Universidad Autónoma de Madrid
Escuela politécnica superior

Doble Grado en Ingeniería Informática y en Matemáticas

Trabajo Fin de Grado

INTRODUCCIÓN AL APRENDIZAJE POR REFUERZO: PROBLEMA BANDIDO MULTIBRAZO

Autor: Álvaro Guinea Juliá
Tutor: Ana González Marcos

Junio 2016

INTRODUCCIÓN AL APRENDIZAJE POR REFUERZO: PROBLEMA BANDIDO MULTIBRAZO

Autor: Álvaro Guinea Juliá
Tutor: Ana González Marcos

Grupo de la EPS (Grupo Aprendizaje Automático)
Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2016

Resumen

Resumen

El objetivo de este trabajo fin de grado es presentar el problema del bandido multibrazo, así como desarrollar una aplicación práctica basada en el problema del bandido multibrazo y destinada al diseño de carteras.

En una primera parte de la memoria realizamos una introducción al aprendizaje por refuerzo, en el que se está incluido el problema del bandido multibrazo. En esta fase se revisa la importancia de la exploración y de la explotación y se introducen conceptos de relevancia, como por ejemplo el rechazo.

En la segunda parte de la memoria desarrollamos el concepto de bandidos estocásticos y presentamos varios algoritmos que se enmarcan dentro de esta idea, como son los algoritmos ϵ-*Greedy*, *Softmax*, UCB1 y otros. Para cada algoritmo se ha realizado un análisis exhaustivo de su funcionamiento.

Por último aplicaremos los algoritmos presentados anteriormente, como técnicas de selección de acciones en carteras financieras. Nos apoyaremos en conocimientos básicos de matemática financiera para conseguir el objetivo de disminuir el riesgo, entendido éste como la varianza de los retornos de cada acción.

Adicionalmente, se ha explorado algoritmos alternativos para el diseño de carteras. Por ejemplo, el algoritmo *Orthogonal Bandit Porfolio*[6] en el cual se eligen carteras ortogonales donde en cada una de ellas se invierte en distintas acciones y con distintos pesos.

Los valores de las acciones se han extraído del entorno *Yahoo Finance*. A partir de los datos capturados se han construido cuatro grupos de pruebas formados por acciones que se encuentran en los índices bursátiles *Standard & Poor's* e IBEX35.

Palabras Clave

Bandido Multibrazo, Bandido Estocástico, UCB, Exploración, Explotación, Cartera, Bandido Adversario, Muestreo de *Thompson*, *Softmax*, Aprendizaje por Refuerzo, Rechazo.

Abstract

The aim of this final degree project is to present the multi-armed bandit problem, as well as develop an application based on the multi-armed bandit problem, which is used in portfolio design.

On the first part of this project, we do an introduction to reinforcement learning, in which is included the multi-armed bandit problem. On this phase we review the importance of explotation and exploration. New relevance concepts are introduced, for example the regret.

On the second part, we develop the concept of stochastic bandits and we present several algorithms that fall on this idea, as for example the algorithms: ϵ-Greedy, Softmax, UCB1 and others. For each algorithm, it has made an exhaustive analisys of its behavior.

By last, we apply the algorithms that we presented before, for selecting assets in a portfolio. We use basic knowledge of mathematical finance for reducing the risk, which is the variance of the return of each asset.

In Addition we explore some alternative algorithms for portfolio design. For example, the algorithm Orthogonal Bandit Porfolio [6], in which orthogonal portfolios are choosen, where in each of them we invest the money in different assets with different weights.

The values of the assets are taken from Yahoo Finace. We use these values to create four test groups, which are formed by stocks from the indexes Standard & Poor's and IBEX35.

Key words

Multi-armed Bandits, Stochastic Bandits, UCB, Explotation, Exploration, Portfolio, Adversarial Bandits, Thompson Sampling, Softmax, Reinforcement Learning, Regret.

Agradecimientos

A mis padres y a mi hermano Víctor, por estar ahí siempre.

A mi compañero de prácticas durante toda la carrera, por haber superado juntos los malos momentos.

A mi tutora Ana González Marcos por su apoyo prestado y dedicación.

Índice general

Índice de Figuras

Índice de Tablas

Glosario de acrónimos

- **UCB**: Upper Confident Bound
- **OBP**: Orthogonal Bandit Portfolio
- **MVP**: Minimun Variance Portfolio
- **EWP**: Equally Weighted Portfolio

1 Introducción

1.1. Introducción: El Aprendizaje por refuerzo

El aprendizaje por refuerzo como dicen Sutton y Barto[1], consiste en aprender que debemos hacer para maximizar una recompensa numérica. Al algoritmo no se le dice que acciones debe tomar, si no que el algoritmo debe probar las acciones para descubrir que acciones dan la mayor recompensa.

El aprendizaje por refuerzo es diferente del aprendizaje supervisado. El aprendizaje supervisado consiste en aprender a partir de una serie de ejemplos dados por un supervisor externo, este método no es adecuado cuando aprendemos mediante interacción. En los problemas interactivos, en los que nuestras acciones generan los resultados, se hace muy complicado encontrar ejemplos, que se puedan usar en el aprendizaje supervisado.

En el aprendizaje por refuerzo, nos encontramos en un entorno incierto, en el que no sabemos nada y la única información que recibimos del entorno esta dada por las recompensas. Podemos entender el aprendizaje por refuerzo como un forma de aprendizaje sin profesor, en el que al alumno no se le dice que tiene que hacer, el único feedback que recibe son las notas que obtiene de los exámenes que realiza.

1.2. El Bandido Multibrazo

El problema del bandido multibrazo fue establecido por Robbins[2] en 1952. Este consiste en un número determinado K de máquinas tragaperras. En cada tiempo t el jugador debe elegir una de las K máquinas, la cual le devolverá una recompensa. El objetivo del jugador será maximizar la recompensa acumulada, es decir encontrar la mejor máquina(aquella que de un mayor número de recompensas altas) en el menor tiempo posible. Las máquinas tragaperras en el contexto del bandido multibrazo, se denominan también bandidos o brazos.

Podría ocurrir que un bandido no óptimo esté dando muchos premios haciéndonos pensar que es el mejor bandido. También podría ocurrir que el mejor bandido no esté dando apenas premios haciéndonos pensar que no es una buena elección en ese momento.

En algún punto podemos llegar a encontrar un bandido que reparta bastantes premios y nos

surja la siguiente pregunta: ¿Seguimos con este bandido o tratamos de encontrar uno mejor?. Este es el dilema Exploración *versus* Explotación (*Exploration vs. Exploitation*).

Para cada tiempo t cada brazo tiene asociada una recompensa $X_{1,t}, ..., X_{K,t}$. Por tanto se hace necesario un algoritmo A, el cual se encargará de elegir un brazo en cada tiempo t. Posteriormente el brazo elegido nos devolverá una recompensa, el jugador sólo podrá observar la recompensa del brazo que haya elegido. Supongamos que el jugador juega un número T de veces y que existen K bandidos, entonces el problema se podría definir como:

Algorithm 1 El Bandido Multibrazo

```
function BandidoMultibrazo(K, T)
    for t=1,2,...T do
        El entorno genera un vector de recompensas {X_{1,t}, ..., X_{K,t}}
        A elige el brazo I_t ∈ {1...K}
        El brazo I_t devuelve una recompensa X_{I_t,t}
        El algoritmo A observa la recompensa obtenida
        El entorno no revela las otras recompensas
    end for
end function
```

La cuestión ahora es, ¿cómo medimos el éxito de nuestro algoritmo?. Para ello definimos el rechazo acumulado o *cumulative regret*:

$$R_T(A) = \max_{i=1,...,K}\{\sum_{t=1}^{T} X_{i,t}\} - \sum_{t=1}^{T} X_{I_t,t}.$$

Fijémonos que el *cumulative regret* compara, el brazo que para un número de T rondas tiene la mayor recompensa acumulada con la recompensa acumulada por el algoritmo A. Nuestro objetivo será elegir el mayor número de veces ese brazo y por tanto minimizar el *cumulative regret*.

Como en general tanto la selección de los brazos por parte del algoritmo, como la generación de recompensas van a seguir procesos aleatorios conviene hablar del rechazo promedio o *averaged regret*. Del cual tenemos dos nociones:

El rechazo acumulado esperado o *expected cumulative regret*:

$$\mathbb{E}[R_T(A)] = \mathbb{E}[\max_{i=1,...,K}\{\sum_{t=1}^{T} X_{i,t}\} - \sum_{t=1}^{T} X_{I_t,t}] = \mathbb{E}[\max_{i=1,...,K}\{\sum_{t=1}^{T} X_{i,t}\}] - \mathbb{E}[\sum_{t=1}^{T} X_{I_t,t}].$$

El otro es el pseudo rechazo o *pseudo regret*:

$$\overline{R_T(A)} = \max_{i=1,...,K}\{\mathbb{E}[\sum_{t=1}^{T} X_{i,t}]\} - \mathbb{E}[\sum_{t=1}^{T} X_{I_t,t}].$$

Fijémonos que el *pseudo regret* es más débil en la noción del *regret* , ya que el *pseudo regret* compara con el brazo que maximiza la esperanza de la recompensa acumulada. Mientras que el *expected regret* es directamente la esperanza del *regret*. Dicho de otro modo:

$$\overline{R_T(A)} \leq \mathbb{E}[R_T(A)].$$

El brazo óptimo según el *pseudo regret* es aquel que maximiza la esperanza de la recompensa acumulada. Todas las definiciones de rechazo aquí expuestas han sido dadas por Bubeck y Cesa-Bianchi[3].

Otras formas menos ortodoxas que el *regret* que usa John Myles[4] a la hora de medir la eficacia de un algoritmo son:

1. La probabilidad de seleccionar el mejor brazo, en cada tiempo t contamos el número de veces que ha sido seleccionado el brazo óptimo y dividimos entre el número de simulaciones. Nuestro objetivo será conseguir que la probabilidad de elegir el brazo óptimo sea igual a uno.

2. La recompensa media, es decir para cada tiempo t sumamos las recompensas obtenidas de todas las simulaciones para ese tiempo t y dividimos entre el número de simulaciones.

3. La recompensa acumulada, suma de las recompensas obtenidas hasta el tiempo t y divididas entre el número de simulaciones. Es una medida muy importante y es interesante contrastarla con la probabilidad de elegir el mejor brazo. Supongamos dos algoritmos A y B. El algoritmo A puede tener una mayor probabilidad de elegir el mejor brazo al final de la simulación, sin embargo B puede tener mayor recompensa acumulada para un mismo horizonte. Esto pone de manifiesto la importancia del horizonte a la hora de seleccionar el algoritmo.

Nuestro problema se reduce a dos puntos fundamentales:

Exploración. El entorno sólo revela la recompensa elegida por el jugador, por lo tanto el jugador tiene que ganar información del entorno tirando repetidamente de todos los brazos.

Explotación. Cada vez que elegimos un brazo que no es el óptimo esto afecta a nuestro *regret*. El objetivo del jugador es minimizar el *regret* eligiendo el brazo óptimo repetidas veces.

1.3. Algunas aplicaciones prácticas

Algunas de las aplicaciones prácticas más importantes son:

1. Desarrollo de pruebas clínicas más éticas[5]. Generalmente cuando se realizan pruebas clínicas se dividen a los pacientes en varios grupos, en los que se prueban varios medicamentos. Tradicionalmente estos grupos permanecen estáticos a lo largo de toda la prueba. Se puede aplicar el problema del bandido multibrazo, durante el desarrollo de la prueba, para desplazar gente al grupo en el que se prueba el medicamento que aparentemente es más eficaz.

2. Diseño de Cartera de acciones[6]. Se puede usar el bandido multibrazo para seleccionar diferentes activos financieros, que incluiremos en una cartera.

3. Recomendaciones personalizadas, por ejemplo de música, películas, artículos[7]. Se usan bandidos contextuales, aquellos que utilizan la información del entorno, para generar recomendaciones personalizadas.

4. Uso del Bandido multibrazo, en vez de A/B *testing* en el desarrollo web[4].

2

Bandidos Estocásticos

2.1. Introducción

Los bandidos estocásticos[3] son aquellos en los que las recompensas son generadas conforme a una función de distribución de probabilidad. Es decir cada brazo $i \in \{1...K\}$, tiene asignada una función de distribución ν_i acotada en [0,1] con una esperanza μ_i. Además las recompensas de un brazo con respecto al tiempo $X_{i,t} \backsim \nu_i$ son independientes e idénticamente distribuidas(i.i.d). Es decir la recompensa dada por el brazo i en el tiempo t no depende de las recompensas anteriores dadas por ese mismo brazo.

Suponiendo un número K de máquinas o brazos y un horizonte T el algoritmo puede escribirse:

Algorithm 2 El Bandido Multibrazo Estocástico

1: **function** BandidoEstocastico(K, T)
2: **for** t=1,2,...T **do**
3: El algoritmo A elige el brazo $I_t \in \{1...K\}$
4: Dado I_t el entorno genera una recompensa $X_{I_t,t} \backsim \nu_{It}$ que es independiente de las recompensas anteriores
5: **end for**
6: **end function**

En el entorno estocástico el *pseudo regret* se puede escribir:

$$R_T = T\mu^* - \sum_{t=1}^{T} \mu_{I_t} = T\mu^* - \mathbb{E}[\sum_{t=1}^{T} \mu_{I_t}].$$

Donde:

$$\mu^* = \max_{i=1,...,K} \{\mu_i\}.$$

$$i^* = \arg\max_{i=1,...,K} \{\mu_i\}.$$

El número de veces que el brazo i ha sido elegido en el horizonte T puede escribirse:

$$N_i(T) = \sum_{t=1}^{T} \mathbb{I}\{I_t = i\}.$$

Fijémonos en que:

$$T = \sum_{i=1}^{K} N_i(T).$$

Además:

$$\mathbb{E}[\sum_{t=1}^{T} \mu_{I_t}] = \mathbb{E}[\sum_{i=i}^{k} N_i(T)\mu_i] = \sum_{i=i}^{k} \mathbb{E}[N_i(T)]\mu_i.$$

Luego el *regret* finalmente se puede expresar:

$$R_T = T\mu^* - \sum_{i=i}^{k} \mathbb{E}[N_i(T)]\mu_i = \sum_{i=i}^{k} \mathbb{E}[N_i(T)]\Delta_i.$$

Donde:

$$\Delta_i = \mu_{i^*} - \mu_i.$$

Así pues siempre que elijamos un brazo óptimo el coste en el *pseudo regret* será cero.

2.2. Bandido Bayesiano

Davidson-Pilon[8] crea este algoritmo de la siguiente manera, supongamos que disponemos K máquinas tragaperras, las cuales devuelven dos tipos de recompensas ceros o unos. Si la máquina devuelve un 1 el jugador ha ganado y se devuelve un cero ha perdido. En este sentido podemos decir que las recompensas siguen una distribución de Bernuilli con una probabilidad p desconocida.

$$X_{i,t} \backsim Ber(p_i)$$

$$X_{i,t} = \begin{cases} 1, & \text{con probabilidad } p_i \\ 0, & \text{con probabilidad } 1 - p_i \end{cases}$$

Supongamos que un determinado brazo i es elegido un numero N de veces. Como las recompensas son independientes de las recompensas anteriores y son idénticamente distribuidas por una Bernuilli de probabilidad p_i. La suma de esas N recompensas del brazo i siguen una Binomial. Es decir: $Y_i = \sum_{j=1}^{N} X_{i,j}$.

$$Y_i \backsim Bi(p_i, N)$$

$$P(Y_i = y) = \binom{N}{y} p_i^y (1 - p_i^{N-y}).$$

Lo que queremos ahora es expresar la función de masa de la Binomial como un función de p, para construir una función de distribución a priori, con la que después aplicaremos el Teorema de Bayes, $f(p) \propto p^a(1-p)^b$.Podemos decir que esa función es proporcional a p elevado al número de ganancias (a), multiplicada por (1-p) elevado al número de perdidas (b). Queremos hacer de

f(p) una función de densidad, por lo tanto debemos usar una constante normalizadora que haga que, $\int_0^1 f(p) \cdot dp$=1.

$$\int_0^1 p^a(1-p)^b dp = \int_0^1 p^{\alpha-1}(1-p)^{\beta-1} dp = B(\alpha, \beta).$$

Donde B es la función Beta de Euler. De tal forma que $\alpha - 1$ es el número de éxitos y $\beta - 1$ el número de fracasos. Por tanto f(p) quedaría como, $f(p) = \frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha,\beta)}$. Luego f(p) sigue una distribución Beta de parámetros α y β.

Ahora usamos el Teorema de Bayes para calcular la probabilidad a posteriori de p.

Theorem 1 (Teorema de Bayes)**.** *Dada una distribución a priori $P(\theta)$, una función de verosimilitud $P(x|\theta)$, y una función de los datos P(x), la probabilidad a posteriori queda:*

$$P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)} = \frac{P(x|\theta)P(\theta)}{\int P(x|\theta')P(\theta')d\theta'}. \tag{2.1}$$

Suponemos que p sigue una distribución a priori dada por la distribución Beta, usemos el Teorema de Bayes para calcular la distribución a posteriori.

$$P(p = x|a, b) = \frac{P(a, b|p = x)P(p = x)}{\int_{y=0}^{y=1} P(a, b|p = y)P(p = y)dy} = \frac{\binom{a+b}{a} x^a(1-x)^b x^{\alpha-1}(1-x)^{\beta-1}/B(\alpha,\beta)}{\int_{y=0}^{y=1} \binom{a+b}{a} y^a(1-y)^b y^{\alpha-1}(1-y)^{\beta-1}/B(\alpha,\beta))dy} =$$

$$= \frac{x^{a+\alpha-1}(1-x)^{b+\beta-1}}{\int_{y=0}^{y=1} y^{a+\alpha-1}(1-y)^{b+\beta-1}dy} = \frac{x^{a+\alpha-1}(1-x)^{b+\beta-1}}{B(\alpha+a, \beta+b)}.$$

Luego la función a posteriori sigue también una distribución Beta. Tenemos por tanto, que la distribución a priori es, $f(p) \backsim Beta(\alpha, \beta)$, y al distribución a posteriori, $f(p|a, b) \backsim Beta(\alpha + a, \beta + b)$, donde a es el número de éxitos y b el número de fracasos. Cuando las distribuciones a priori y a posteriori de un mismo parámetro pertenecen a la misma familia de distribuciones, se denominan distribuciones conjugadas. Para pasar de la distribución a priori a la posteriori tan sólo tenemos que sumar a α el número de éxitos y a β el número de fracasos.

Cabe destacar que la función Beta de Euler $B(x, y) = \int_0^1 t^{x-1}(1-y)^{y-1}dt$ cuando x e y son enteros, la función se puede escribir como, $B(x, y) = \frac{(x-1)!(y-1)!}{(x+y-1)!}$. Mencionar también que cuando α y β son ambas iguales a 1, la distribución Beta es igual a una distribución uniforme.

Volviendo al problema del bandido multibrazo y aplicando lo anterior construimos el siguiente algoritmo:

Algorithm 3 Bandido Bayesiano1

function BANDIDOBAYESIANO1(K, T)
  Para cada brazo i $\in \{1...K\}$ fijamos $S_i = 0$ y $F_i = 0$
  for t=1,2,...T **do**
    Para cada brazo i $\in \{1...K\}$ $B_i \backsim Beta(1 + S_i, 1 + F_i)$
    A elige el brazo $I_t = \arg\max_{1...K} B_i$
    El brazo I_t devuelve una recompensa $X_{I_t,t}$
    if $X_{I_t,t}$==1 **then**
      $S_{I_t} = S_{I_t} + 1$
    else
      $F_{I_t} = F_{I_t} + 1$
    end if
  end for
end function

S_i y F_i es el número de éxitos y fracasos de cada brazo i, respectivamente. Este algoritmo también se denomina muestreo de Thompson o Thompson sampling.

2.2.1. Resultados

Ahora probemos nuestro algoritmo con tres bandidos que siguen tres distribuciones de Bernuilli con probabilidades 0.6, 0.75 y 0.8. En la Figura 2.1 se puede observar como evolucionan las distribuciones Betas a posteriori de los tres bandidos, a medida que el algoritmo avanza en el tiempo. Conforme el parámetro β se hace más grande la distribución Beta de desplaza hacia la izquierda, cuando α se hace más grande la distribución se desplaza hacia la derecha. Como podemos observar, el algoritmo a medida que avanza el tiempo se va acercando cada vez más al mejor bandido, aquel que tiene una probabilidad de 0.85.

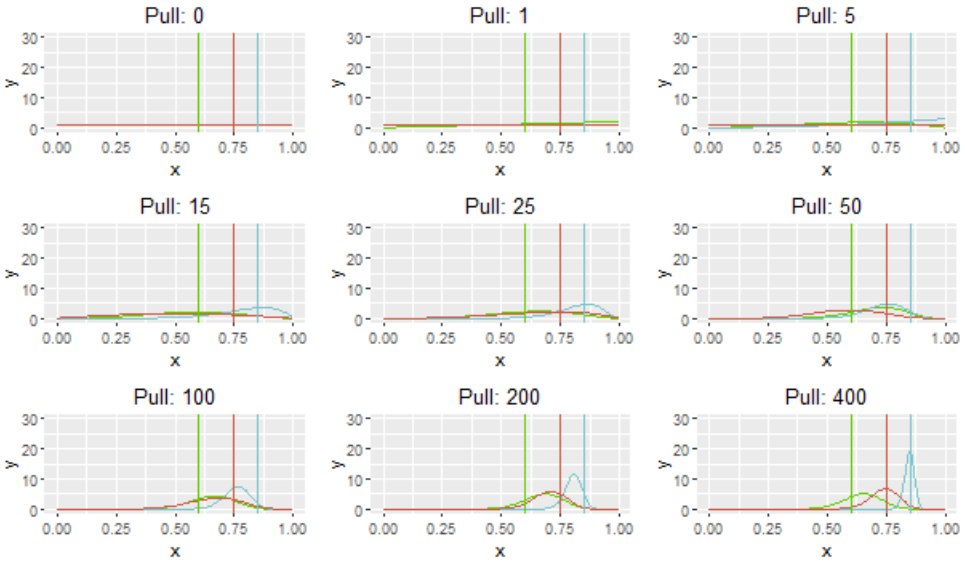

Figura 2.1: Evolución de las distribuciones Beta a posteriori a lo largo de la simulación. El eje x representa el valor de p, el eje y representa el valor de la distribución.

En la siguiente Figura 2.2 se muestra el rechazo del bandido bayesiano para el mismo problema que en la Figura 2.1, en el cual se ha elegido un horizonte de 5000 *pulls* o tiradas y se ha simulado 25 veces. Podemos observar como al principio el *regret* aumenta muy rápido porque nos encontramos todavía en la fase de exploración a medida que avanza el tiempo nuestro algoritmo detecta cual es el mejor bandido y empieza a explotarlo y por tanto el *regret* crece de manera más lenta.

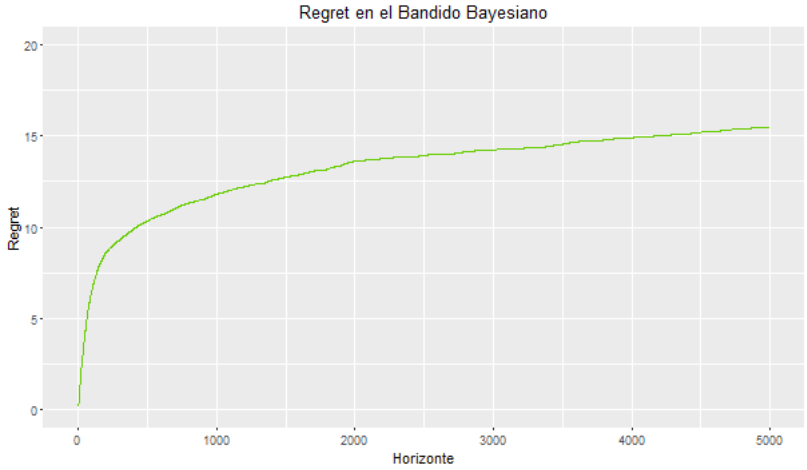

Figura 2.2: Rechazo obtenido sobre 25 simulaciones, con un horizonte de hasta 5000 unidades de tiempo. El eje x representa el horizonte y el eje y el valor del rechazo.

2.2.2. Generalización a todo tipo de bandidos estocásticos

Uno de los problemas de la solución propuesta, es que sólo funciona con brazos que siguen distribuciones de Bernuilli. Una opción para utilizar las muestras de Thompson para cualquier tipo de bandido estocástico, que recoge Davidson-Pilon[8] es su libro es la siguiente. Supone que el brazo i sigue una distribución desconocida acotada en [0,1] con una esperanza μ_i. Este brazo en cada tiempo t devuelve una recompensa $\overline{X_{i,t}}$. Como la recompensa se encuentra entre [0,1] podemos usar dicha recompensa para realizar un prueba de Bernuilli con probabilidad $\overline{X_{i,t}}$. Esta prueba nos devuelve una recompensa $X_{i,t}$ que es la que utilizaremos para actualizar las distribuciones *Betas* que se encargan de seleccionar los brazos. El algoritmo quedaría de la forma:

Algorithm 4 Bandido Bayesiano2

function BANDIDOBAYESIANO2(K, T)
  Para cada brazo i $\in \{1...K\}$ fijamos $S_i = 0$ y $F_i = 0$
  for t=1,2,...T **do**
    Para cada brazo i $\in \{1...K\}$ $B_i \backsim Beta(1 + S_i, 1 + F_i)$
    A elige el brazo $I_t = \arg\max_{1...K} B_i$
    El brazo I_t devuelve una recompensa $\overline{X_{i,t}}$
    Realizamos una prueba de Bernulli con probabilidad $\overline{X_{i,t}}$ lo que nos da una recompensa $X_{I_t,t}$
    if $X_{I_t,t}$==1 **then**
      $S_{I_t} = S_{I_t} + 1$
    else
      $F_{I_t} = F_{I_t} + 1$
    end if
  end for
end function

Se puede ver que $P(X = 1) = \int_0^1 x f_i(x) dx = \mu_i$ donde f_i es la función de densidad desconocida para el brazo i.

2.3. Algoritmo ε-greedy

El algoritmo ε-greedy[1] es uno de los algoritmos más simples. En cada ronda t elegimos con probabilidad 1-ε el brazo que tenga mejor media empírica y con probabilidad ε un brazo aleatorio,con $\varepsilon \in [0, 1]$. De tal forma que la probabilidad de coger un brazo i en el tiempo t+1 sería:

$$P_i(t+1) = \begin{cases} 1 - \varepsilon + \varepsilon/K, & \text{si i= } \arg\max_{j=1,\dots,K} \widehat{\mu_j}(t) \\ \varepsilon/K, & \text{en otro caso} \end{cases}$$

Donde $\widehat{\mu_j}(t)$ es la media empírica de las recompensas obtenidas del brazo j en el tiempo t. Si elegimos un $\varepsilon = 1$ nuestro algoritmo elegirá siempre de forma aleatoria entre las diferentes opciones, exploración pura. Si elegimos $\varepsilon = 0$ elegirá siempre el brazo con mejor media empírica pero no explorará nunca otras opciones, explotación pura. Cuanto más grande sea el ε más tiempo va a dedicar nuestro algoritmo a la exploración. Suponiendo un número K de Bandidos, un horizonte T el algoritmo puede escribirse como:

Algorithm 5 $\epsilon - Greedy$

function GREEDY(K, T, ε)
  for t=1,2,...T **do**
    A selecciona el brazo $i = \arg\max_{j=1,\dots,K} \widehat{\mu_j}(t)$ con probabilidad $1-\varepsilon$ o elige un brazo aleatorio con probabilidad ε
    El entorno devuelve la recompensa para el brazo elegido por A
    Actualizamos el valor de la media empírica del brazo elegido con la recompensa devuelta
  end for
end function

2.3.1. Resultados

A continuación probamos el algoritmo con diferentes valores para epsilon (0.1, 0.2, 0.3, 0.4 y 0.5). En un primer ejemplo (Figura 2.3) probamos como funciona el algoritmo con tres bandidos que siguen una Bernuilli con probabilidades de 0.1, 0.1 y 0.9, con un horizonte de 250 y 500 simulaciones. En la Figura 2.3 se muestra la probabilidad de elegir el mejor bandido es decir, de elegir el bandido que devuelve 1 con una probabilidad de 0.9. Podemos observar que cuanto menor es el epsilon más tiempo tarda en averiguar (aprender) cual es el mejor bandido, esto es así porque cuanto menor sea el epsilon menos explora y por tanto más tarda en aprender. Si el epsilon es mayor encuentra el bandido óptimo antes, ya que explora más pero al cabo de un tiempo esta exploración deja de ser beneficiosa, ya que debería ponerse a explotar.

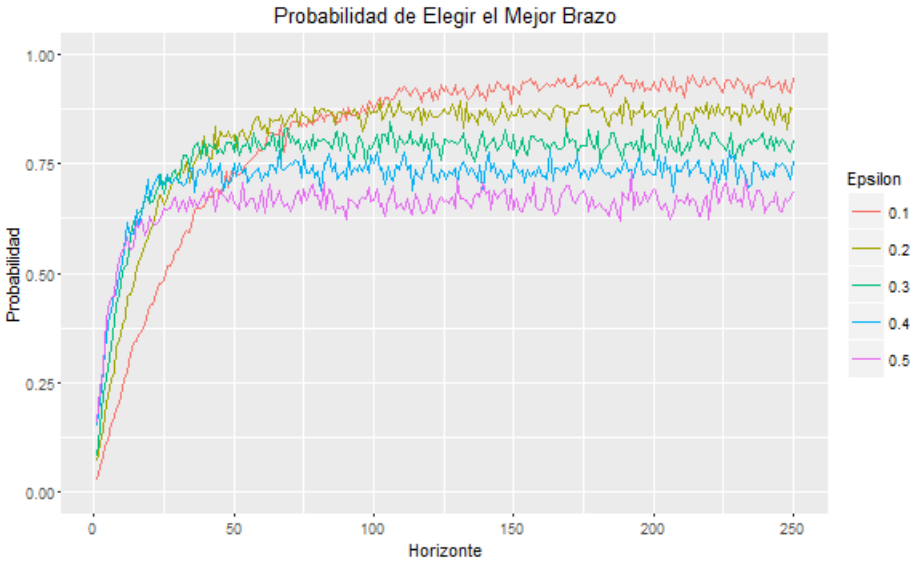

Figura 2.3: Resultados $\varepsilon - Greedy$ con tres bandidos Bernuilli con probabilidades 0.1, 0.1 y 0.9. En el eje de la x se representa el horizonte y en el eje de la y la probabilidad de elegir el bandido óptimo

Podríamos preguntarnos, ¿qué ocurriría si los valores de las probabilidades de los bandidos fuesen más similares?. La Figura 2.4 muestra las probabilidades de elegir el mejor bandido cuando la probabilidades de los bandidos son 0.1, 0.2 y 0.3 se simula 500 veces con un horizonte de 2000. Fijémonos en que al ser las probabilidades de nuestros bandidos similares al algoritmo le cuesta mucho más aprender. Mientras que en la Figura 2.3 con un horizonte de 250 nuestro algoritmo con ε=0.1 ya sabía cual era el mejor bandido. Como se puede apreciar en la Figura 2.4 para el valor ε=0.1 con un horizonte de 250, el algoritmo todavía esta aprendiendo.

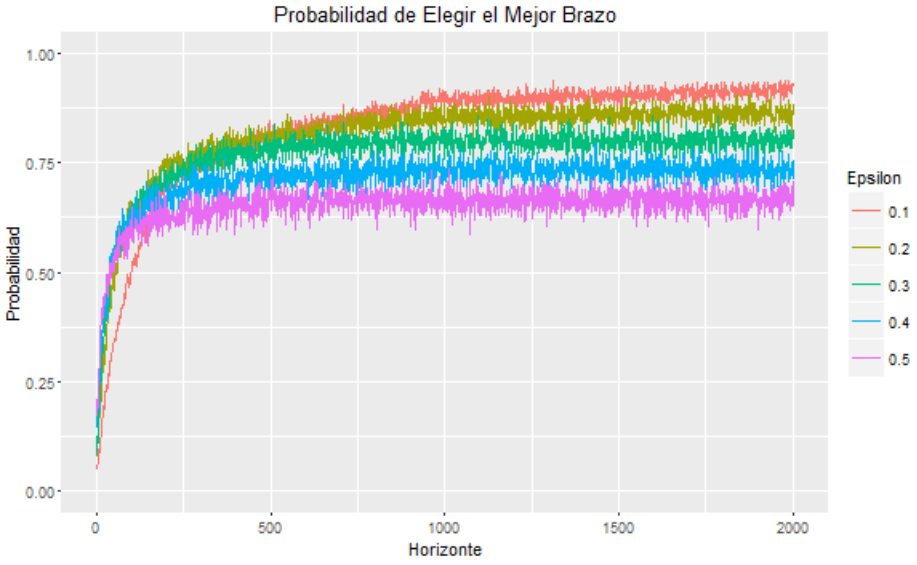

Figura 2.4: Resultados $\varepsilon - Greedy$ con tres bandidos Bernuilli con probabilidades 0.1, 0.2 y 0.3. En el eje x se representa el horizonte y en el eje de la y se representa la probabilidad de elegir el brazo óptimo.

Vamos a ver qué ocurre ahora si los valores son similares pero muy altos (Figura 2.5), en este

caso probamos con bandidos con probabilidades de 0.7, 0.8 y 0.9, que simulamos 500 veces con un horizonte de 500. Podemos observar que aprende más lento que en el primer ejemplo, pero más rápido que en el segundo ejemplo. Esto es así, porque nuestro algoritmo se basa en la media empírica a la hora de seleccionar el brazo, por tanto al tener probabilidades altas devuelve 1 muy a menudo sobre todo con el bandido con probabilidad 0.9, provocando que haya más exploración.

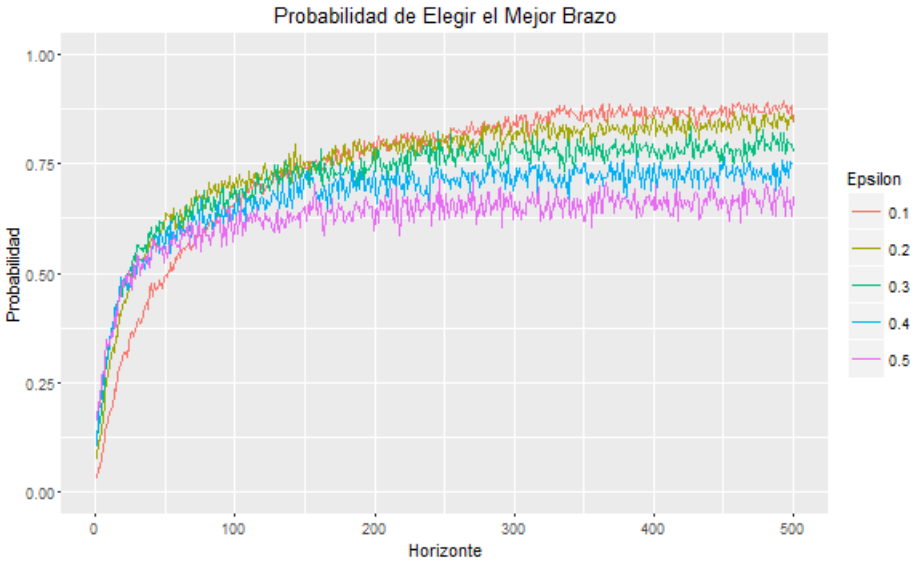

Figura 2.5: Resultados $\varepsilon - Greedy$ con tres bandidos Bernuilli con probabilidades 0.7, 0.8 y 0.9. En el eje x se representa el horizonte y en el eje de la y se representa la probabilidad de elegir el brazo óptimo.

En estos ejemplos hemos podido observar que el horizonte influye en el desarrollo de los algoritmos. Así, para un horizonte pequeño conviene usar un epsilon grande (explorar más). Sin embargo si el horizonte es largo conviene un epsilon pequeño.

2.4. Softmax

Uno de los problemas que nos ofrece el algoritmo $\varepsilon - Greedy$ es que cuando explora, todos los brazos tienen la misma probabilidad de ser elegidos[4]. Si por ejemplo, dos bandidos devuelven recompensas uno 10 % de las veces y otro un 13 % es necesario explorar más, para asegurarnos de que el brazo que elegimos es el correcto. En otro caso, tenemos que uno devuelve recompensas un 90 % y otro un 10 %, no es necesario explorar tanto como en el caso anterior, porque entonces perderíamos dinero.

Por tanto, se hace necesario un algoritmo en el que la probabilidad de seleccionar un brazo es proporcional a su media empírica. De esta manera los brazos con mayor media empírica, tendrán más probabilidad de ser elegidos. En el algoritmo Softmax[1] la probabilidad de elegir el brazo i es:

$$P_i(t+1) = \frac{e^{\widehat{\mu_i}(t)/\tau}}{\sum_{j=1}^{K} e^{\widehat{\mu_j}(t)/\tau}}$$

Donde $\widehat{\mu_j}(t)$ es la media empírica de las recompensas obtenidas del brazo j en el tiempo t y τ es el parámetro de control de la temperatura (hace referencia a su uso en la mecánicas

estadísticas), que controla la aleatoriedad de las elecciones. Así cuanto menor sea τ más peso da a la media empírica a la hora de elegir los brazos por tanto explora menos. Cuanto mayor sea τ menos peso da a las medias empíricas y más va a explorar.

Si $\tau = 0$ es explotación pura, elige el brazo que tenga mayor media empírica. Mientras que si $\tau \longrightarrow \infty$ el algoritmo se comporta como una distribución uniforme, exploración pura. Suponiendo un número K de Bandidos y un horizonte T, el algoritmo Softmax puede escribirse:

Algorithm 6 SoftMax

function SoftMax(K, T, τ)
  for t=1,2,...T **do**
    Para todo brazo i calculamos $P_i(t) = \frac{e^{\widehat{\mu_i}(t-1)/\tau}}{\sum_{j=1}^{K} e^{\widehat{\mu_j}(t-1)/\tau}}$
    Elegimos el brazo I_t conforme a la distribución anterior
    El brazo I_t devuelve una recompensa $X_{I_t,t}$
    Con la recompensa obtenida $X_{I_t,t}$ actualizamos $\widehat{\mu_{I_t}}(t)$
  end for
end function

2.4.1. Resultados

Al igual que hicimos en el $\varepsilon - Greedy$ probamos primero con tres distribuciones Bernuilli donde las probabilidades de dar una recompensa están muy separadas. En este caso probamos con tres Bernuillis con probabilidades 0.1, 0.1 y 0.9 que simulamos 500 veces con un horizonte de 250 (Figura 2.6). Podemos observar que para τ=0.1 prácticamente desde el principio el algoritmo descubre cual es el mejor y su probabilidad es prácticamente 1 al final de la simulación. Como podemos observar, este algoritmo es bastante mejor que el Greedy con ε=0.1 para el mismo caso. En este caso τ=0.1 es un buen valor para cuando las tasas de recompensas de los bandidos tienen valores alejados, sin embargo veremos que para valores cercanos, τ=0.1 ofrece resultados más adversos.

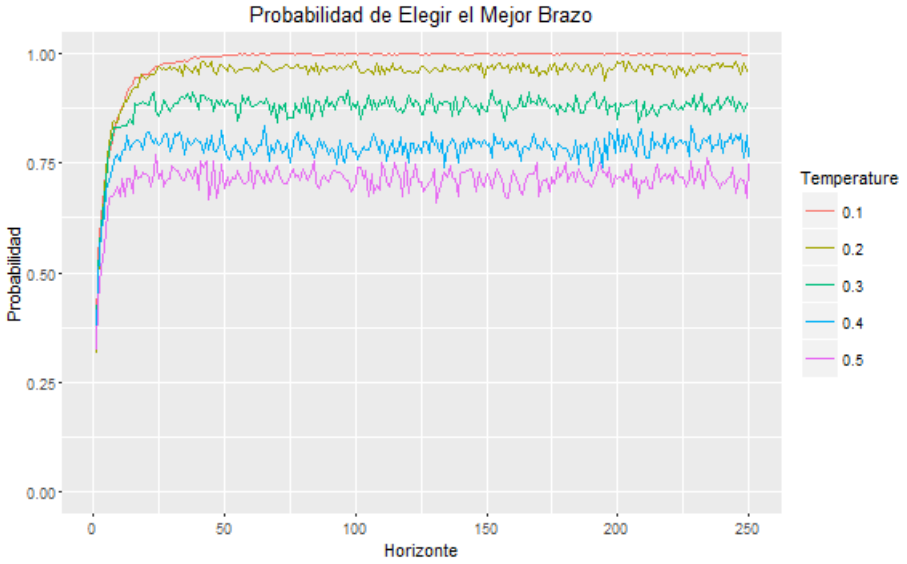

Figura 2.6: Resultados SoftMax con tres bandidos Bernuilli con probabilidades 0.1, 0.1 y 0.9. En el eje x se representa el horizonte y en el eje de la y se representa la probabilidad de elegir el brazo óptimo.

Veamos ahora que ocurre cuando los valores de las tasas de recompensas de los brazos son similares. Probamos tres Bernuillis con probabilidades 0.1, 0.2 y 0.3 (Figura 2.7). Lo simulamos 500 veces con un horizonte de 250, para unos valores de τ iguales a los del ejemplo anterior(Figura 2.6). Los resultados obtenidos difieren mucho del ejemplo anterior. En este caso τ=0.1 sigue dando el mejor resultado aunque la probabilidad de elegir el mejor brazo no esta cerca del 1.

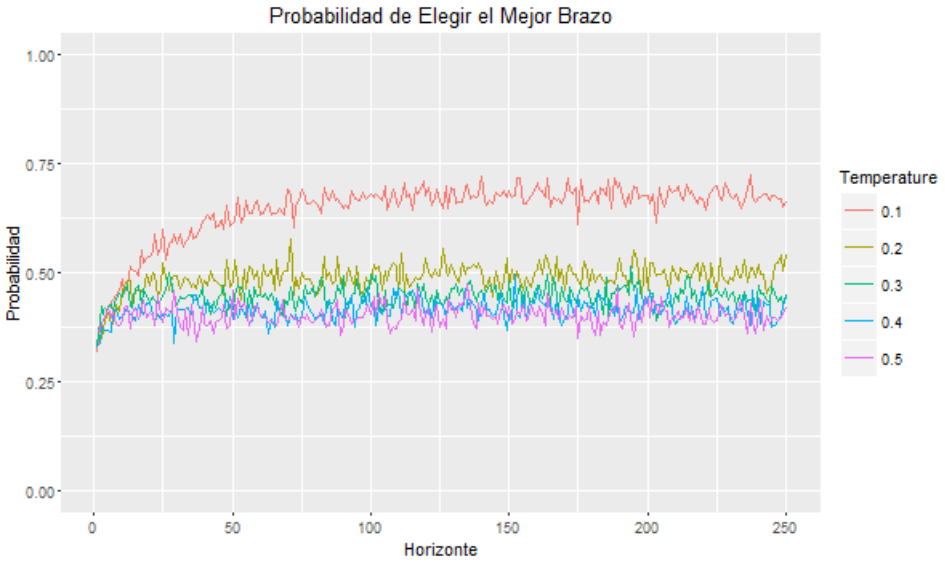

Figura 2.7: Resultados SoftMax con tres bandidos Bernuilli con probabilidades 0.1, 0.2 y 0.3. En el eje x se representa el horizonte y en el eje de la y se representa la probabilidad de elegir el brazo óptimo.

Ahora veamos qué sucede con valores de τ más bajos (Figura 2.8) de esta manera daremos más peso a la media empírica de cada brazo. Con ello la probabilidad de elegir el mejor brazo aumenta, pero disminuirá la exploración y su crecimiento será menor (es decir necesitaremos un horizonte mayor para obtener mejor resultados). Podemos ver que para τ=0.05 nos da mejores resultados que τ=0.1 pero explora menos que el anterior y por tanto crece de manera más lenta. Merece la pena observar τ=0.03, para este horizonte se obtienen peores resultados que con τ=0.1 y τ=0.05 si el horizonte aumenta se obtienen buenos resultados. La explicación es que explora mucho menos y por tanto el crecimiento es lento.

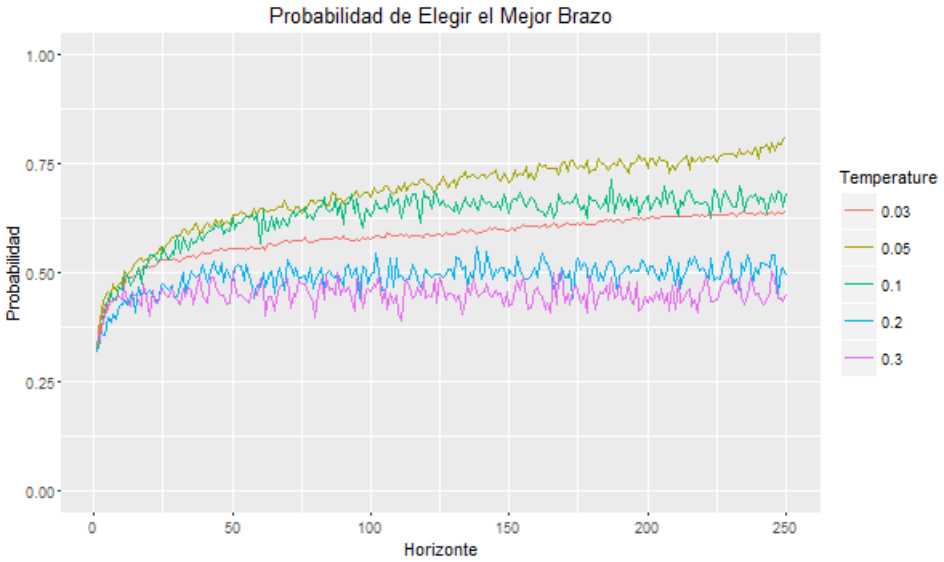

Figura 2.8: Resultados SoftMax con tres bandidos Bernuilli con probabilidades 0.1, 0.2 y 0.3. En el eje x se representa el horizonte y en el eje de la y se representa la probabilidad de elegir el brazo óptimo.

Sigamos con el mismo ejemplo, sólo que esta vez hemos aumentado el horizonte primero a 2000 (Figura 2.9) y después a 5000 (Figura 2.10). En estos ejemplo podemos ver que para los valores de τ=0.05 y τ=0.03, los resultado ofrecidos por el algoritmo mejoran bastante, aunque como se ha explicado en el caso anterior crecen de manera lenta.

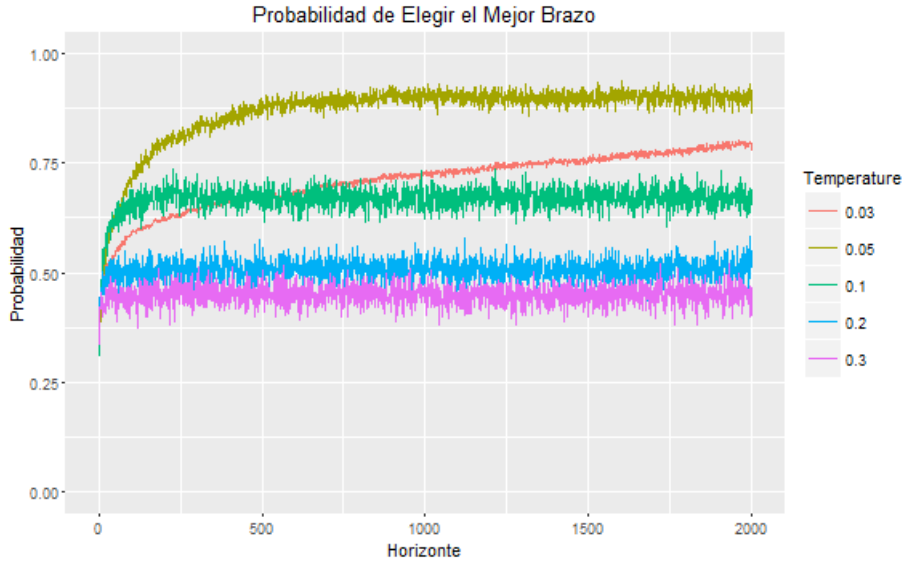

Figura 2.9: Resulados SoftMax con tres bandidos Bernuilli con probabilidades 0.1, 0.2 y 0.3. En el eje x se representa el horizonte y en el eje de la y se representa la probabilidad de elegir el brazo óptimo.

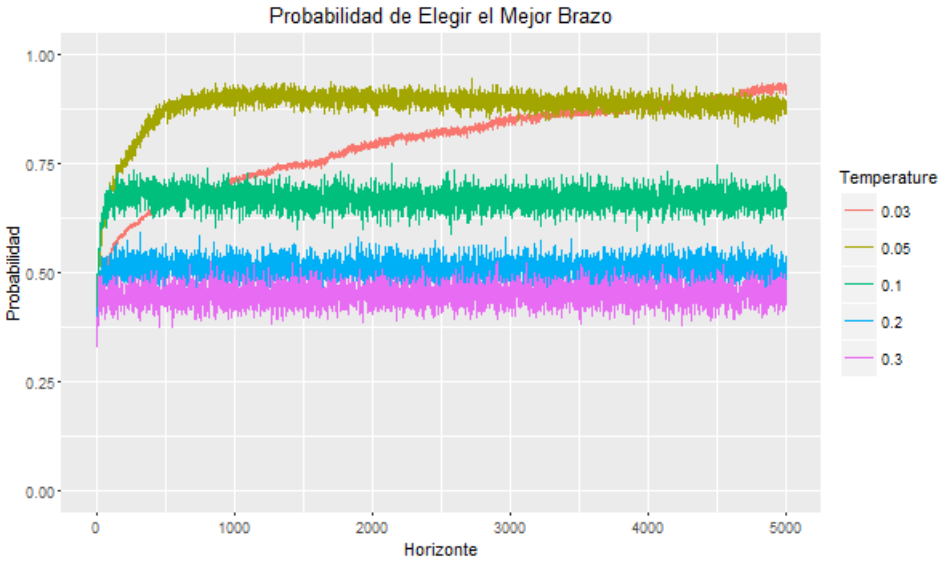

Figura 2.10: Resultados SoftMax con tres bandidos Bernuilli con probabilidades 0.1, 0.2 y 0.3. En el eje x se representa el horizonte y en el eje de la y se representa la probabilidad de elegir el brazo óptimo.

Pero,¿por qué ocurre esto?. ¿Por qué necesitamos valores de τ pequeños, cuando los valores de las tasas de recompensas están muy juntas, para que la probabilidad de elegir el mejor brazo converja a 1?. Sigamos con el mismo ejemplo y supongamos que el algoritmo se ha lanzado un número infinito de veces y que cada brazo ha sido elegido un número extraordinariamente alto. Por la Ley de los Grandes Números podemos decir que las medias empíricas de recompensas de cada uno de los brazos convergen a la esperanza de cada una de las distribuciones. De tal forma que para τ=0.1 la probabilidad de elegir el mejor brazo sería aproximadamente: $\frac{e^3}{e^3+e^2+e^1} \approx 0.665241$. Sin embargo para τ=0.03 la probabilidad quedaría: $\frac{e^{10}}{e^{10}+e^{6,6667}+e^{3,3333}} \approx 0.9643687$.

En este ejemplo cabe destacar también la recompensa acumulada (Figura 2.11). Mientras que en la probabilidad de elegir el mejor brazo, hacia el final del horizonte, parece ser que el parámetro $\tau = 0{,}03$, es la mejor aproximación, ver Figura 2.10. Si miramos la recompensa acumulada (Figura 2.11), podemos observar que en este caso τ=0.05 es la mejor elección, pues nos da la mayor recompensa acumulada. En el caso de que el horizonte fuese mucho mayor resultará mejor elegir τ=0.03. Por tanto, el horizonte influye, a la hora de seleccionar los parámetros que más nos benefician.

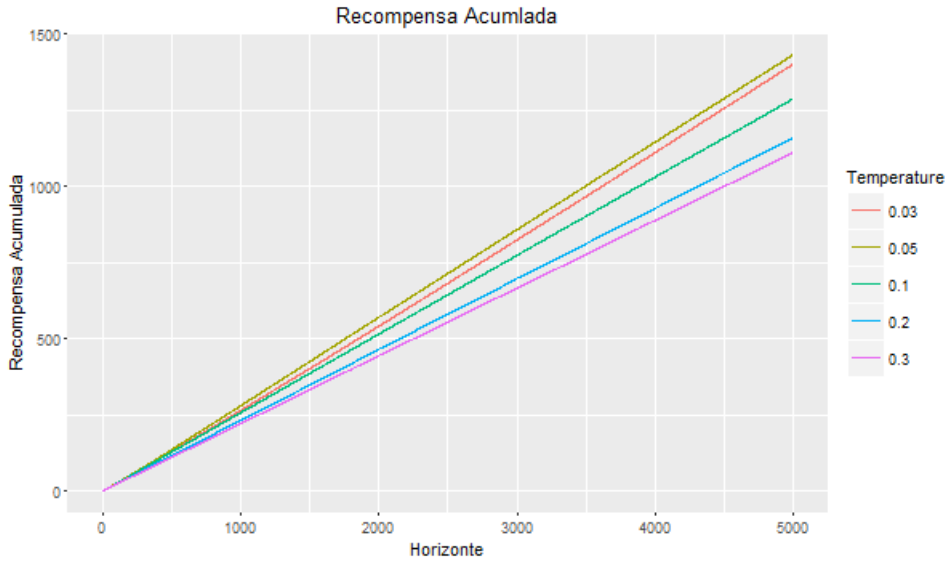

Figura 2.11: Resultados SoftMax con tres bandidos Bernuilli con probabilidades 0.1, 0.2 y 0.3. En el eje x se representa el horizonte y en el eje de la y se representa la recompensa acumulada.

En el caso de que las tasas de recompensas sean 0.7, 0.8 y 0.9 es un caso muy similar al anterior. Debido a que las probabilidades son muy altas no es necesario usar τ tan pequeñas. Las figuras, Figura 2.12, Figura 2.13, Figura 2.14 y Figura 2.15 representan simulaciones con distinto horizonte y con distintos valores del parámetro τ. Figura 2.12 corresponde a τ grandes, representan *tau* pequeÃśo y el horizonte va aumentando. Esta serie de figuras demuestra la importancia que tiene el horizonte. Por ejemplo en la Figura 2.15, se puede apreciar cómo los valores de τ=0.1 y τ=0.09 superan al resto cuando el horizonte es grande. Valores pequeños τ=0.05 colapsan y no mejoran con el horizonte.

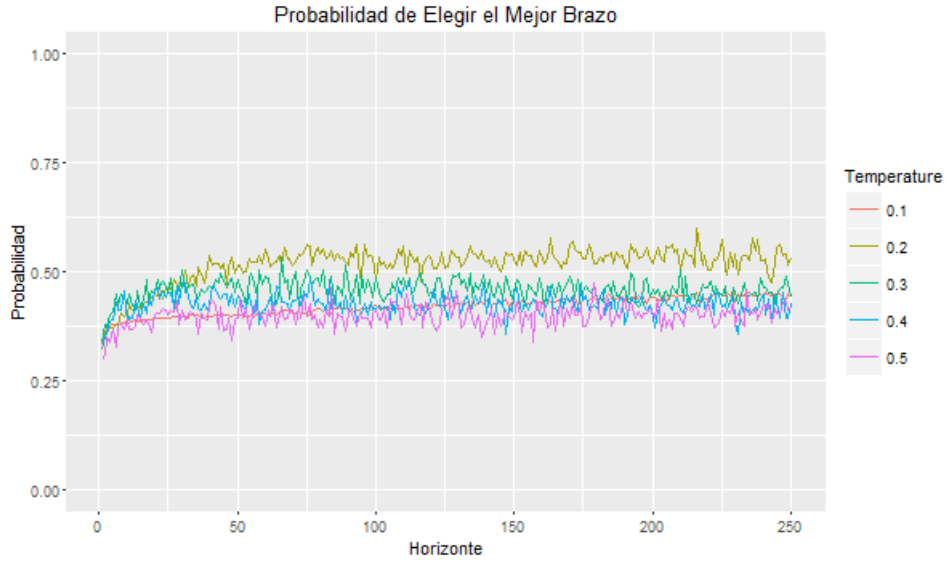

Figura 2.12: Resultados SoftMax con tres bandidos Bernuilli con probabilidades 0.7, 0.8 y 0.9. En el eje x se representa el horizonte y en el eje de la y se representa la recompensa acumulada.

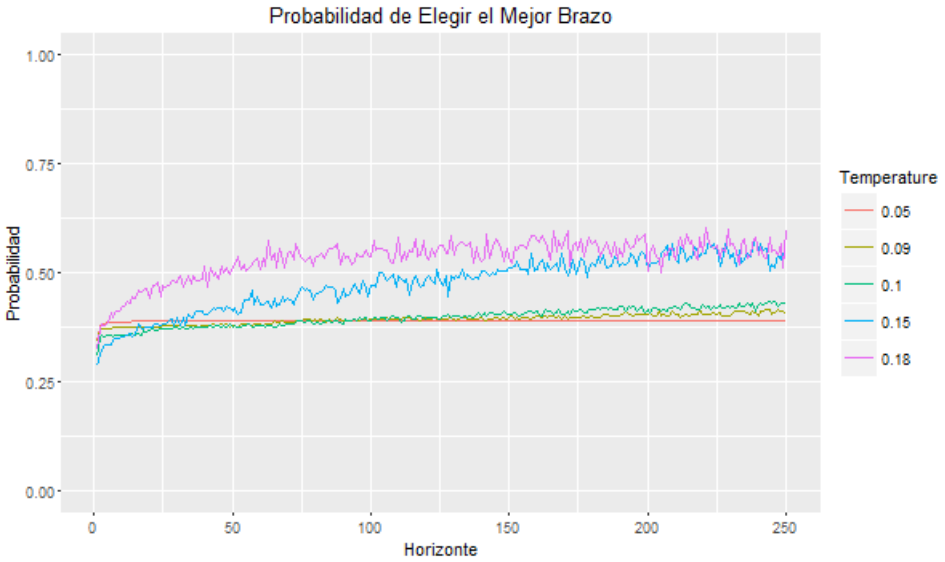

Figura 2.13: Resultados SoftMax con tres bandidos Bernuilli con probabilidades 0.7, 0.8 y 0.9. En el eje x se representa el horizonte y en el eje de la y se representa la recompensa acumulada.

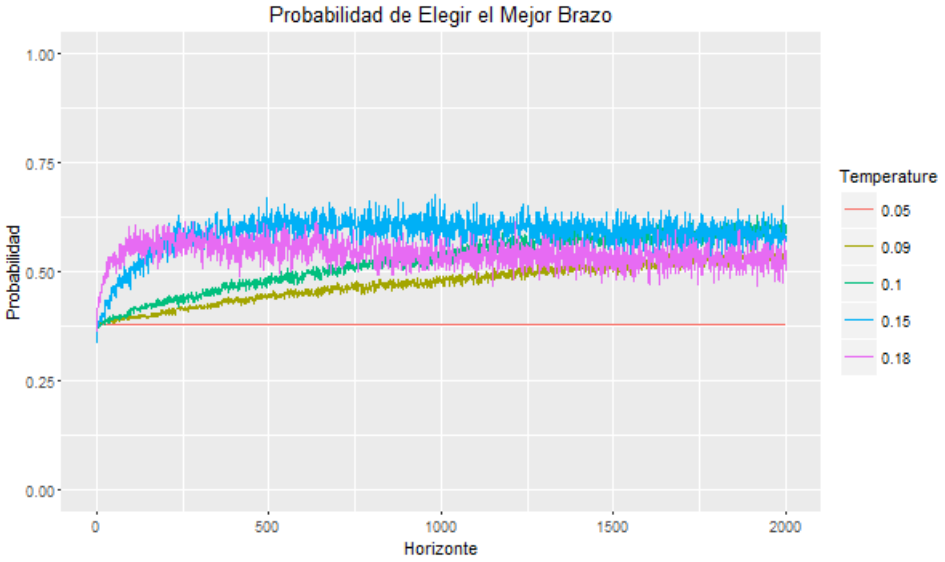

Figura 2.14: Resultados SoftMax con tres bandidos Bernuilli con probabilidades 0.7, 0.8 y 0.9. En el eje x se representa el horizonte y en el eje de la y se representa la recompensa acumulada.

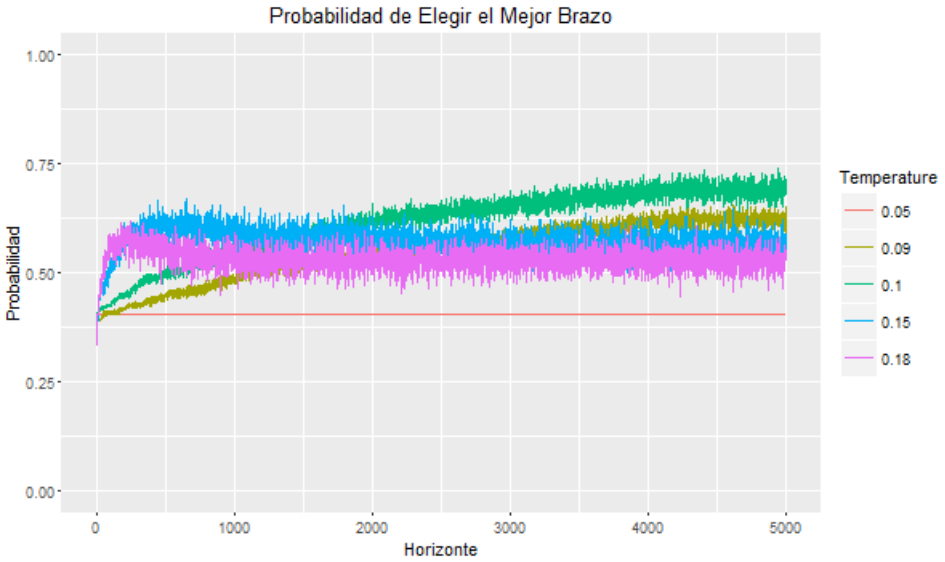

Figura 2.15: Resultados SoftMax con tres bandidos Bernuilli con probabilidades 0.7, 0.8 y 0.9. En el eje x se representa el horizonte y en el eje de la y se representa la recompensa acumulada.

2.4.2. *Anneling* o Enfriamiento

Hemos visto que es conveniente explorar al principio, es decir recabar información para saber dÃşnde se encuentra el mejor bandido, para después pasar a explotar, es decir rentabilizar los datos obtenidos en la exploración. No sirve de mucho encontrar rápidamente al mejor bandido si después no podemos explotarlo. El algoritmo que aquí desarrollamos es una variación del algoritmo dado por John Myles[4].

En términos de los parÃąmetros τ y ε consistirá en que estas variables sean muy grandes al principio de la simulación y conforme el tiempo vaya avanzado se disminuya el valor de τ y del ε. De tal forma que al final la probabilidad de elegir el mejor bandido vaya convergiendo a 1. Este método se denomina enfriamiento, ya que tiene ciertas similitudes con la física de partículas, cuanto mayor sea la temperatura mayor movimiento tienen las partículas (mayor exploración). El movimiento de las partículas decrece a medida que disminuye la temperatura.

Un forma de conseguir este efecto es ir actualizando los parámetro ε y τ conforme avanza el tiempo. Al principio se dan valores altos, para irlos reduciendo a medida que avanza el tiempo. Una forma de conseguirlo sería hacer lo siguiente:

$$\varepsilon(t+1) = (1-\alpha)\varepsilon(t)$$

$$\tau(t+1) = (1-\alpha)\tau(t)$$

Donde $\alpha \in (0,1)$, $\varepsilon(0) = 1 y \tau(0) = 1$, de tal forma que al inicio de nuestra simulación tendremos una distribución uniforme. El parámetro α determina como de rápido decrece ε o τ. Si α es cercano a 1 pasará de explorar a explotar rápidamente. Mientras que si es cercano a 0 el algoritmo hará la transición de exploración a explotación de manera más gradual. Hay que tener cuidado con elegir valores muy grandes para el α, ya que como pasa de la fase de exploración a la de explotación muy rápido, puede cometer equivocaciones a la hora de elegir el bandido.

Los resultados obtenidos en 500 simulaciones con el *Anneling SoftMax* mejoran significativamente, con respecto al SoftMax (Figura 2.16). La Figura 2.16 consta de tres gráficas distintas,

la superior izquierda muestra los resultados obtenidos con tres bandidos Bernuilli con probabilidades 0.1, 0.1 y 0.9. La gráfica de la derecha superior muestra los resultados obtenidos con tres bandidos Bernuilli con probabilidades 0.1, 0.2 y 0.3. Y la gráfica inferior muestra los resultados obtenidos con tres bandidos Bernuilli con probabilidades 0.7, 0.8 y 0.9. En el caso de los dos últimos gráficos sólo es necesario un horizonte de 500 para que la probabilidad de elegir el mejor brazo sea aproximadamente 1. Podemos observar que en el primer gráfico las constantes de aprendizaje son más grandes esto es así porque las diferencia que hay entre las diferentes tasas de recompensas son muy grandes (0.1 y 0.9) por lo que nos es necesario explorar tanto. Si embargo en los otros dos gráficos si que es necesario ya que la diferencia entre las diferentes tasas de recompensas es menor y necesitamos que el cambio de exploración a explotación se realice de manera más gradual.

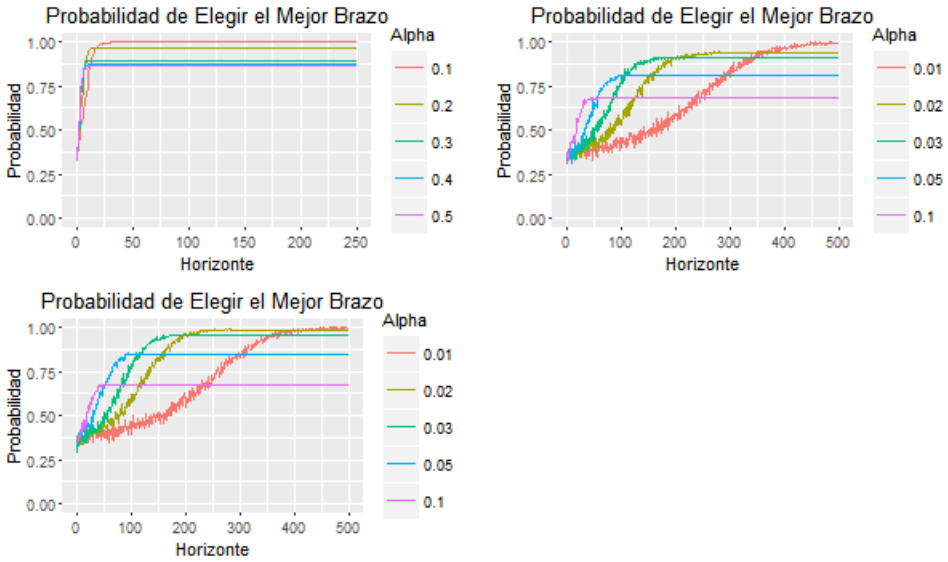

Figura 2.16: Simulación de Anneling SoftMax para los tres ejemplos que hemos visto anteriormente. La gráfica superior izquierda muestra los resultados obtenidos con tres bandidos Bernuilli con probabilidades 0.1, 0.1 y 0.9. La gráfica de la derecha superior muestra los resultados obtenidos con tres bandidos Bernuilli con probabilidades 0.1, 0.2 y 0.3. Y la gráfica inferior muestra los resultados obtenidos con tres bandidos Bernuilli con probabilidades 0.7, 0.8 y 0.9.

Los resultados para el Anneling del $\varepsilon - Greedy$, también mejoran significativamente para los mismos ejemplos con respecto al $\varepsilon - Greedy$ (Figura 2.17).

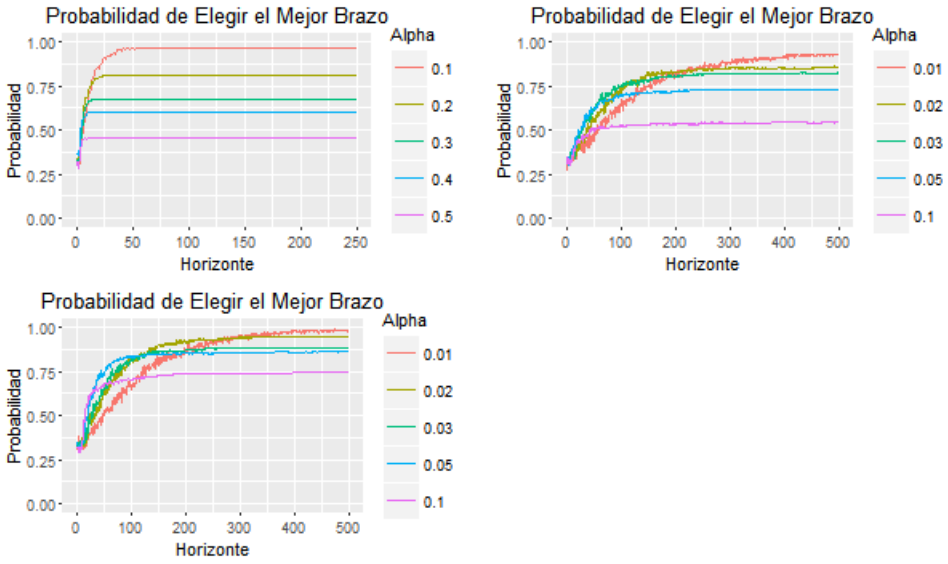

Figura 2.17: Simulación de Anneling Greedy para los tres ejemplos que hemos visto anteriormente. La gráfica superior izquierda muestra los resultados obtenidos con tres bandidos Bernuilli con probabilidades 0.1, 0.1 y 0.9. La gráfica de la derecha superior muestra los resultados obtenidos con tres bandidos Bernuilli con probabilidades 0.1, 0.2 y 0.3. Y la gráfica inferior muestra los resultados obtenidos con tres bandidos Bernuilli con probabilidades 0.7, 0.8 y 0.9.

2.4.3. Valores Iniciales Optimistas

Otra opción mas sencilla que la anterior para conseguir un mayor exploración inicial, consiste en asignar valores iniciales optimistas a las medias empíricas de las recompensas $\widehat{\mu_j}(t)$, por ejemplo asignar $\widehat{\mu_j}(0)$=0.7 $\forall j$. De esta forma los brazos son probados varias veces antes de converger a su esperanza. A mayor valor inicial, mayor tiempo se va a dedicar el algoritmo a la exploración.

A continuación mostramos como este sencillo método mejora un $\varepsilon-$Greedy con ε=0.1 a la hora de elegir entre tres bandidos Bernuillis con probabilidades 0.1,0.1 y 0.9. Simulamos 500 veces con un horizonte de 250 (Figura 2.18).

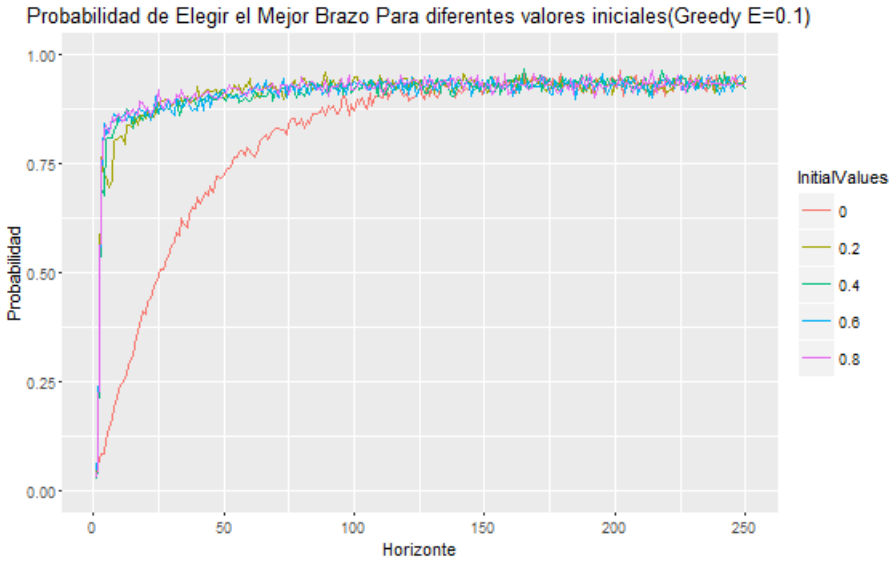

Figura 2.18: A
lgoritmo ϵ-Greedy con ε=0.1 con valores iniciales optimistas para tres bandidos Bernuilli con probabilidades 0.1, 0.1 y 0.9. En el eje x se representa el horizonte y en el eje de la y se representa la probabilidad de elegir el brazo óptimo.

La utilización de valores iniciales optimistas puede ser realizada por otros algoritmos que explicamos a continuación: Pursuit, UCB, Exp3, Comparación por refuerzo.

2.5. Pursuit

Este algoritmo[1] utiliza las medias empíricas de las recompensas de cada brazo para actualizar las probabilidades de elegir cada brazo, llamadas preferencias. Las preferencias se actualizan de dos modos diferentes. Si un brazo tiene la mayor media empírica entonces la preferencia aumentará. Si no, la preferencia del brazo se actualizará disminuyendo su valor.

La idea es que al principio todas las preferencias tengan el mismo valor, es decir sigan una distribución uniforme y conforme el tiempo vaya transcurriendo, la probabilidad de elegir el bandido óptimo tienda a 1. Para ello usaremos un constante de aprendizaje que determinará como de rápido aprendemos, o dicho de otra manera, como de rápido pasamos de exploración a explotación. La constante también determina la cantidad de riesgo que nuestro algoritmo incorpora a la hora de tomar decisiones. A una constante mayor, mayor riesgo y por tanto mayor probabilidad de que el algoritmo se equivoque. Este procedimiento es muy similar al visto en el algoritmo *Anneling*, en la sección 2.4.2.

La probabilidad(preferencia) del brazo i se actualiza de la siguiente forma:

$$P_i(t) = \begin{cases} P_i(t-1) + \beta(1 - P_i(t-1)), & \text{Si } i = \arg\max_j \widehat{\mu_j}(t) \\ P_i(t-1) + \beta(0 - P_i(t-1)), & \text{En caso contrario} \end{cases}$$

Donde $\beta \in (0, 1)$ es la constante de aprendizaje e inicialmente todos los brazos tiene la misma preferencia $P_i(0) = 1/K$. Se puede observar que aquel brazo que tiene la mayor media empírica, su preferencia aumenta mientras que en otro caso disminuye. La cantidad que disminuye o aumenta viene determinada por la constante de aprendizaje β.

Suponiendo un número K de bandidos, un horizonte T el algoritmo Pursuit puede escribirse como:

Algorithm 7 Pursuit

function PURSUIT(K, T, β)
  $P_i(0) = 1/K \ \forall i \in 1, ..., K$
  for t=1,2,...T **do**
    Elegimos el brazo I_t conforme a las probabilidades dadas por la preferencias $P_i(t-1)$
    El brazo I_t devuelve una recompensa $X_{I_t,t}$
    Con la recompensa obtenida $X_{I_t,t}$, actualizamos $\widehat{\mu_{I_t}}(t)$
    for i=1,...,K **do**
      if i==arg max$_j$ $\widehat{\mu_j}(t)$ **then**
        $P_i(t) = P_i(t-1) + \beta(1 - P_i(t-1))$
      else
        $P_i(t) = P_i(t-1) + \beta(0 - P_i(t-1))$
      end if
    end for
  end for
end function

Los resultados obtenidos para el Pursuit son similares a los obtenidos con el algoritmo *Anneling* (Figura 2.19).

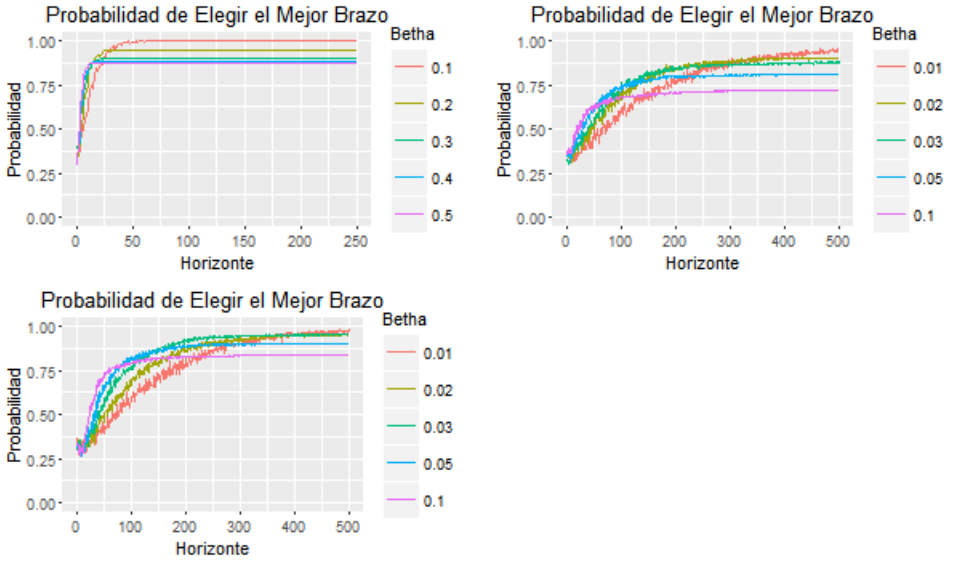

Figura 2.19: Simulación del algoritmo Pursuit para los tres ejemplos que hemos visto anteriormente. La gráfica superior izquierda muestra los resultados obtenidos con tres bandidos Bernuilli con probabilidades 0.1, 0.1 y 0.9. La gráfica de la derecha superior muestra los resultados obtenidos con tres bandidos Bernuilli con probabilidades 0.1, 0.2 y 0.3. Y la gráfica inferior muestra los resultados obtenidos con tres bandidos Bernuilli con probabilidades 0.7, 0.8 y 0.9

2.6. *Reinforcement Comparation*

El algoritmo *Reinforcement Comparation*[1] se basa en, una de las principales intuiciones del aprendizaje por refuerzo: es que las acciones seguidas de grandes recompensas son más probables

que se repitan, mientras que las acciones seguidas de recompensas pequeñas son menos probables que se repitan.

Pero Â£cÃşmo sabemos si una recompensa es grande o pequeña?. Sólo podemos saber si una recompensa es grande o pequeña si la comparamos con un estándar o nivel de referencia que llamamos recompensa de referencia. Una elección natural de esta recompensa podría ser la media de las recompensas recibidas.

Una recompensa se considera alta si es mayor que la recompensa estándar y se considera baja si es menor que la recompensa de referencia. Si las recompensas que obtienen un brazo son mayores que la recompensa estándar o de referencia, su probabilidad de ser elegido aumenta, mientras que si ocurre lo contrario sus probabilidades disminuyen. Esto se denomina Refuerzo por Comparación.

En este algoritmo no mantenemos información sobre las medias empíricas de las recompensas de cada brazo, a diferencia de los anteriores algoritmos que sÃŋ lo hacían.

La probabilidad de seleccionar el brazo i en el tiempo t es:

$$P_i(t) = \frac{e^{\Pi_i(t-1)}}{\sum_{j=1}^{K} e^{\Pi_j(t-1)}}$$

Donde $\Pi_i(t)$ es la preferencia del brazo i en el tiempo t. Inicialmente las preferencias se igualan a cero($\Pi_i(0) = 0$). Si el brazo I_t es seleccionado solo se actualiza la preferencia del brazo I_t, el resto de preferencias se dejan como están. La actualización del brazo I_t se hace de la siguiente forma:

$$\Pi_{I_t}(t) = \Pi_{I_t}(t-1) + \beta(r_{I_t} - \overline{r(t-1)})$$

Donde $\beta \in (0, 1)$ es una constante de aprendizaje, r_{I_t} es la recompensa devuelta por el brazo I_t y $\overline{r(t)}$ es la recompensa de referencia. Esta parte del algoritmo permite una funcionalidad parecida al algoritmo *Pursuit* y al *Anneling*. Si las recompensas que obtiene el brazo son altas con respecto a la recompensa de referencia, la preferencia de dicho brazo subirá y por tanto explotará más que antes. Sin embargo, si obtiene recompensas bajas su preferencia disminuirá y esta vez explorará más en vez de explotar. Una vez actualizada la preferencia del brazo actualizamos su recompensa de referencia.

$$\overline{r(t)} = (1 - \alpha)\overline{r(t-1)} + \alpha r_{I_t}$$

Donde $\alpha \in (0, 1)$ es una constante de aprendizaje que regula el peso que tiene la última recompensa obtenida en la recompensa de referencia. Si es cercana a 1, la última recompensa tiene mucho peso y si es cercana a cero tiene poco peso dentro de la recompensa de referencia. En nuestro algoritmo debemos inicializar la variable referencia en el tiempo t=0 ($\overline{r(0)}$), este valor se puede inicializar de manera optimista o de acuerdo a algún conocimiento a priori que tengamos.

Este algoritmo, a diferencia de los anteriores, en los que se aumentaba la explotación en detrimento de la exploración a lo largo del tiempo, la exploración puede aumentar o disminuir en cualquier momento según las recompensas que se obtengan. Si estas son altas con respecto a la de referencia la exploración disminuye y si son bajas, ocurre lo contrario. Esto permite al *Reinforcement Comparation* adaptarse cuando las recompensas dadas por los brazos no son estacionarias.

Suponiendo un número K de bandidos, un horizonte T el algoritmo de Comparación por Refuerzo puede escribirse como:

Algorithm 8 Reinforcement Comparation

function REINFORCEMENT COMPARATION($K, T, \beta, \alpha, \overline{r(0)}$)
  $\Pi_i(0) = 0 \ \forall i \in 1, ..., K$
  for t=1,2,...T **do**
    Calculamos $\forall i \ P_i(t) = \frac{e^{\Pi_i(t-1)}}{\sum_{j=1}^{K} e^{\Pi_j(t-1)}}$
    Seleccionamos el brazo I_t conforme a las probabilidades $P_i(t)$
    El brazo I_t devuelve una recompensa r_{I_t}
    $\Pi_{I_t}(t) = \Pi_{I_t}(t-1) + \beta(r_{I_t} - \overline{r(t-1)})$
    $\overline{r(t)} = (1-\alpha)\overline{r(t-1)} + \alpha r_{I_t}$
  end for
end function

2.6.1. Resultados

A continuación mostramos los resultados obtenidos para el algoritmo de Refuerzo por Comparación. Primero probaremos el algoritmo dejando el valor α fijo. En un primer ejemplo probamos como funciona el algoritmo con tres bandidos que siguen una Bernuilli con probabilidades de 0.1,0.1 y 0.9, con un horizonte de 250 y 500 simulaciones(Figura 2.20). Además inicializamos de forma óptimista el algoritmo $\overline{r(0)} = 0{,}8$ y fijamos el $\alpha = 0{,}1$. Cuanto mayor β más rápido va a converger la probabilidad a 1 y por tanto antes convergemos al brazo óptimo. Si variamos sólo la β, el algoritmo es similar al pursuit y al anneling.

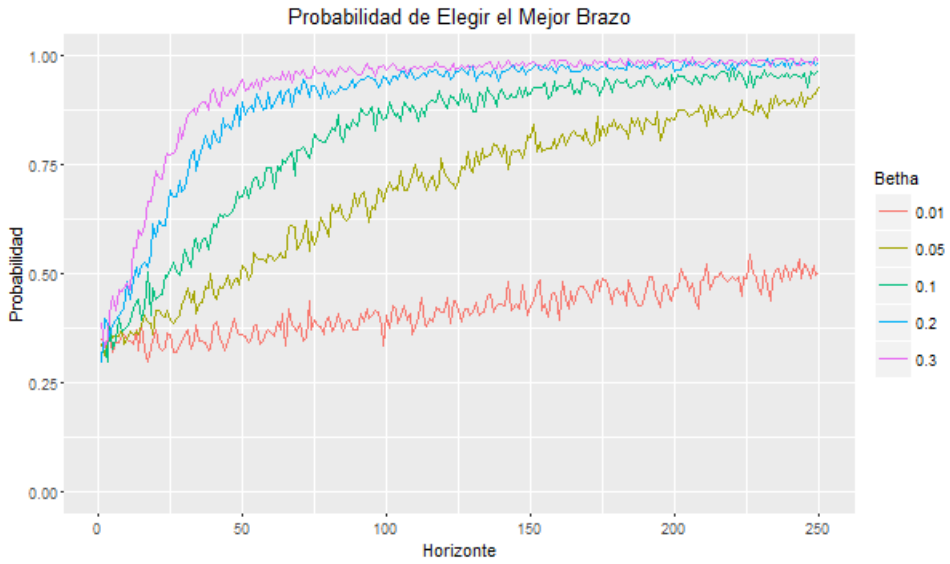

Figura 2.20: Resulados Comparación por refuerzo con α fijo, con tres bandidos Bernuilli con probabilidades 0.1, 0.1 y 0.9. En el eje x se representa el horizonte y en el eje de la y se representa la probabilidad de elegir el brazo óptimo.

A continuación mostramos que ocurre si sólo variamos α y dejamos fija β. Para el mismo ejemplo, con $\beta = 0{,}3$ y $\overline{r(0)} = 0{,}8$ (Figura 2.21), los resultados son muy similares en todos los casos. Ello se debe a que la recompensa es estacionaria, es decir las tasas de recompensas de los bandidos se mantienen constante a lo largo del tiempo. Si bien es cierto que difieren un poco cuando $\alpha = 0{,}01$, debido a que la recompensa de referencia da muy poco peso a la última recompensa obtenida haciendo que la recompensa de referencia converja más lentamente a la

esperanza del bandido óptimo. Esto se observa mejor si las tasas de recompensas de los bandidos son similares (Figura 2.22).

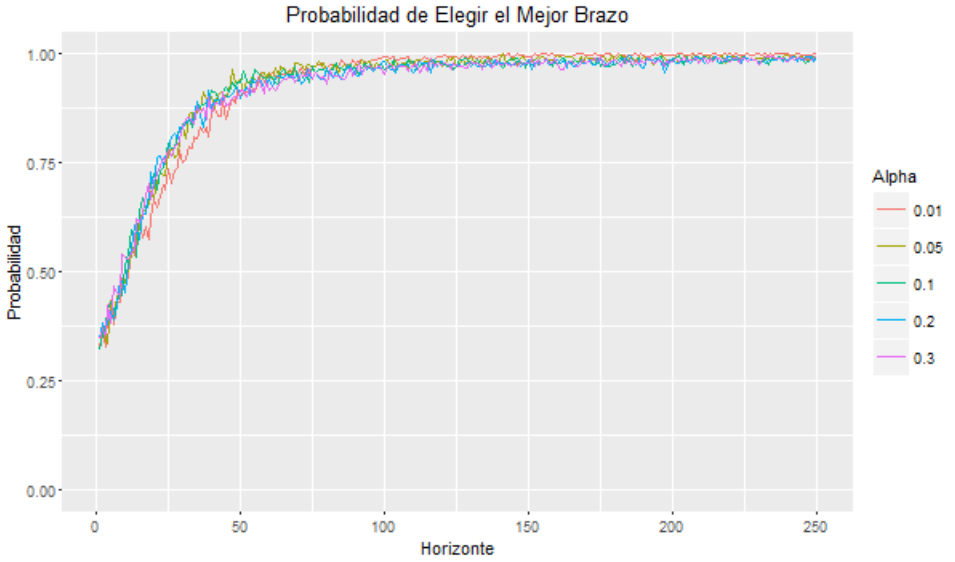

Figura 2.21: Comparación por refuerzo con β fijo, para tres bandidos Bernuilli con probabilidades 0.1, 0.1 y 0.9. En el eje x se representa el horizonte y en el eje de la y se representa la probabilidad de elegir el brazo óptimo.

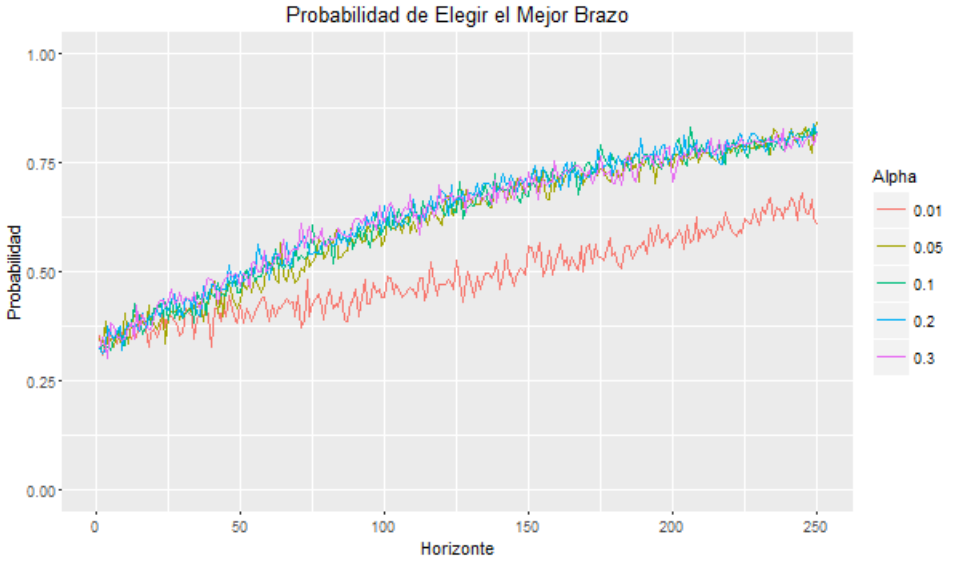

Figura 2.22: Comparación por refuerzo con β fijo, para tres bandidos Bernuilli con probabilidades 0.1, 0.2 y 0.3. En el eje x se representa el horizonte y en el eje de la y se representa la probabilidad de elegir el brazo óptimo.

2.7. UCB1

El algoritmo desarrollado por Auer, Cesa-Bianchi y Fischer[9], presta atención no sólo a lo que ya se sabe, sino a cuÃąnto se sabe. A diferencia de los algoritmos anteriores en los que sólo nos interesaba la recompensada dada por cada uno de los brazos, en este caso vamos a interesarnos por medir la confianza (cuÃąnto sabemos) de los diferentes brazos. Así pues, si sabemos muy poco sobre un brazo, su confianza es baja, y deberemos explorarlo para extraer información sobre él, es decir somos óptimistas con respecto a lo desconocido, pues suponemos que de los brazos de los que no tenemos mucha información pueden dar buenas recompensas y deben ser explorados.

Para medir la confianza, utilizamos intervalos de confianza sobre las medias empíricas de las recompensas de los brazos. La política de selección llevada a cabo por el algoritmo consitirá en elegir aquel brazo que tenga una mayor cota superior de confianza, al fin y al cabo UCB significa *Upper Confidence Bound*. Para construir el intervalo de confianza del algorimo UCB1 usamos la desigualdad de Chernoff-Hoeffding.

Theorem 2 (Desigualdad de Chernoff-Hoeffding)**.** *Sean $X_1, X_2, ..., X_n$ vairables aleaotrias independientes, cuyos valores se encuentran en el intervalo [0,1]. Si $\mu_i = E[X_i]$, $X = 1/n \sum_{i=1}^{n} X_i$, $\mu = E[X] = 1/n \sum_{i=1}^{n} \mu_i$, entonces,*

$$P(|X - \mu| > a) \leq 2e^{-2a^2/n} \tag{2.2}$$

Luego obtenemos dos desigualdades,

$$P(X - \mu > a) \leq e^{-2a^2/n} \tag{2.3}$$

$$P(X - \mu < -a) \leq e^{-2a^2/n} \tag{2.4}$$

Ahora usando la desigualdad anterior (2.4) construiremos un intervalo de confianza:

1. Si n_j es el número de veces que se ha seleccionado el brazo j e igualamos $a = a(j,T) = \sqrt[2]{2log(T)/n_j}$ (a, lo llamamos bonus) y sustituimos en la desigualdad de Chernoff-Hoefding (2.4) ($n = n_j$): $P(X - a > \mu) \leq T^{-4}$. Esta expresiÃşn converge a cero rápidamente, conforme el número de veces que el brazo j ha sido elegido aumenta.

2. Si el brazo j no es elegido, si en la ronda anterior teníamos que a=a(j,T) ahora tendremos $a = a(j, T+1) = \sqrt[2]{2log(T+1)/n_j}$, el valor nuevo de a es mayor quÃľ el correspondiente al tiempo T, y por tanto el intervalo de confianza crece.

3. Sin embargo si el brazo j es elegido, el bonus (a=a(j,T+1)) decrece y por tanto el intervalo de confianza es menor.

Así pues la política de selección de brazos quedaría:

$$\arg\max_{j=1...K} \widehat{\mu}_j + \sqrt[2]{2log(T)/n_j} \tag{2.5}$$

En este caso lo que estamos haciendo es seleccionar el brazo que tenga la máxima cota superior del intervalo de confianza. El algoritmo UCB1 para un horizonte T y un número de brazos K se puede escribir como.

Algorithm 9 UCB1

function UCB1(K, T)
  for t=1,2,...K **do**
    Selecciona el brazo t
  end for
  for t=K+1,K+2,...T **do**
    $I_t = \arg\max_{j=1...K} \widehat{\mu_j} + \sqrt[2]{2log(t)/n_j}$
    Selecciona el brazo I_t
    Con la recompensa obtenida $X_{I_t,t}$ actualizamos $\widehat{\mu_{I_t}}(t)$
  end for
end function

Al principio del algoritmo probamos todos los brazos una vez. Otra diferencia con respecto a los algoritmos que hemos explicado anteriormente es que el UCB1 es determinista.

Una buena noticia es que el *regret* de este algoritmo puede ser acotado:

Theorem 3. *Para todo $K > 1$, si el algoritmo UCB1 es usado en K máquinas, con distribuciones de recompensas $\nu_1, ..., \nu_K$ acotadas en [0,1] con esperanzas $\mu_1, ..., \mu_k$, entonces el regret para un horizonte T es como mucho:*

$$[8 \sum_{i:\mu_i<\mu^*} log(T)/\Delta_i] + (1 + \Pi^2/3)(\sum_{j=1}^{k} \Delta_j) \tag{2.6}$$

La prueba del Teorema 3 se encuentra en el Anexo E.

El resultado dado por el teorema 3, indica que la peor cota para el rechazo del algoritmo UCB1, es logarítimico. Si nos fijamos en la ecuación (2.6) dada por el teorema 3, tiene una primera parte $[8 \sum_{i:\mu_i<\mu^*} log(T)/\Delta_i]$. Esta parte nos indica que vamos a seleccionar un número logarítmico de veces bandidos no óptimos. Si Δ_i es pequeño (es decir, la tasa de recompensas de los bandidos tienen valores próximos), requeriremos seleccionar un mayor número de veces un brazo no óptimo, hasta encontar el mejor brazo. Es decir si la tasas de recompensa de los bandidos están próximas, el algoritmo necesitará explorar más, sin embargo si están alejadas unas de otras necesitará explorar menos. Por otro lado, el segundo término de la ecuación $(6)(1 + \Pi^2/3) * (\sum_{j=1}^{k} \Delta_j)$, es la penalización por elegir un bandido no óptimo, es decir si Δ_i es grande (las tasas de recompensa de los bandidos están alejadas), la penalización por elegir un brazo subóptimo será grande. Es decir si las tasas de recompensas de los bandidos están alejadas unas de otras, la penalización por explotar un brazo subóptimo será mayor, que si las tasas de recompensa están cerca unas de otras.

2.7.1. Resultados

A continuación se muestran las pruebas realizadas para el algoritmo UCB1. En un primer ejemplo probamos como funciona el algoritmo con tres bandidos que siguen una Bernuilli con probabilidades de 0.1, 0.1 y 0.9, con un horizonte de 250 y 500 simulaciones. En la Figura 2.23 se muestra la probabilidad de elegir el mejor bandido, el *regret* del algoritmo y el *worst regret* que nos da el teorema 3. Es interesante comprobar que a medida que la probabilidad de elegir el mejor bandido va convergiendo a 1, el *regret* se va suavizando y crece más lentamente. Otra observación interesante del algoritmo, es que el UCB1 produce mucho ruido, esto es debido a que el UCB1 mide la confianza de cada brazo y de vez en cuando explora aquellos sobre los que tiene poca información. Por tanto el algoritmo UCB1 siempre va a explorar cada cierto tiempo.

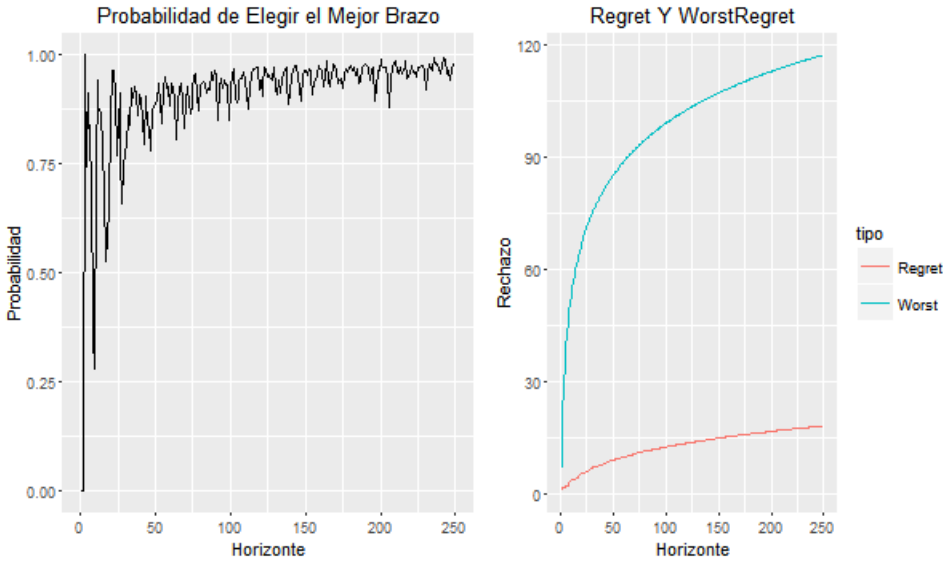

Figura 2.23: Resulados UCB con tres bandidos Bernuilli con probabilidades 0.1, 0.1 y 0.9. La gráfica de la izquierda representa la probabilidad de elegir el brazo óptimo, y la gráfica de la derecha representa el rechazo y el peor rechazo del algoritmo.

En la Figura 2.24 se muestra el comportamiento del algoritmo cuando las tasas de recompensa tiene valores cercanos,simulamos 500 veces con un horizonte de 2000. En estos casos podemos ver como aumenta el *regret* y el *worst regret*, en comparación con el ejemplo anterior (Figura 2.23).

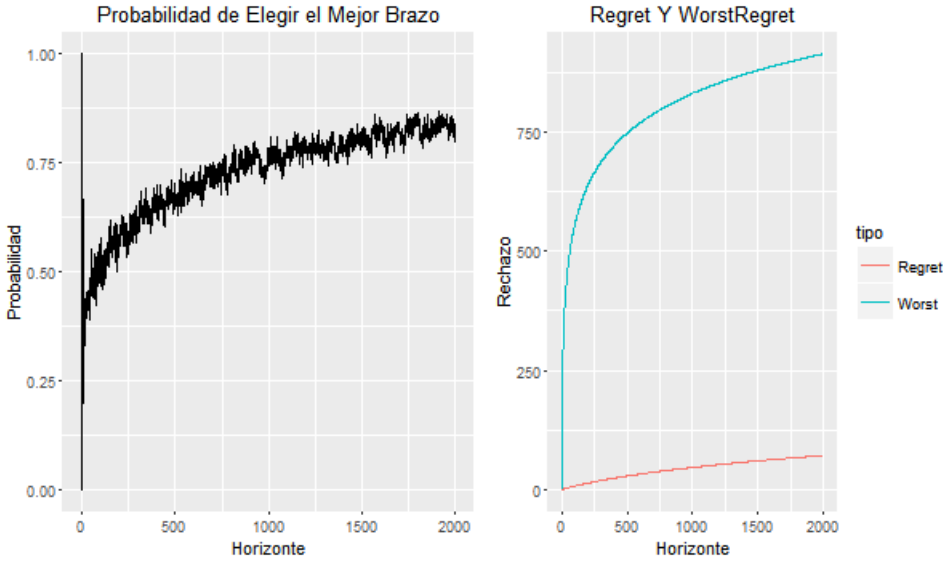

Figura 2.24: Resulados UCB con tres bandidos Bernuilli con probabilidades 0.1, 0.2 y 0.3. La gráfica de la izquierda representa la probabilidad de elegir el brazo óptimo, y la gráfica de la derecha representa el rechazo y el peor rechazo del algoritmo.

3

Aplicación Práctica

3.1. Introducción

A continuación se hace uso de los algoritmos explicados a lo largo del proyecto, con el objetivo de usarlos para la selección de acciones y el diseño de carteras. En general se hará uso de los algoritmos explicados en la sección Bandidos Estocásticos. Podemos considerar cada acción como un brazo, el cual devuelve recompensas cada cierto periodo de tiempo (diariamente, semanalmente, mensualmente ...), estas recompensas son los retornos de cada acción en cada periodo de tiempo. Los retornos de cada acción los calculamos de la siguiente manera: $R_t = \frac{PC_t - PA_t}{PA_t}$. Donde R_t es el retorno de la acción en el periodo t y PC_t es el precio de cierre de la acción en el periodo t y PA_t es el precio de apertura en el periodo t. Esto es equivalente a la rentabilidad obtenida por una unidad monetaria (dollar, euro, ...) por periodo de tiempo.

Sin embargo a la hora de aplicar los algoritmos vistos en las secciones anteriores se plantean una serie de problemas:

1. A diferencia del problema del bandido multibrazo en el que sólo devolvía recompensa el brazo elegido, en este caso todos los brazos devuelven recompensas al finalizar el periodo.
2. Cuando aplicamos los algoritmos vistos en la Sección de Bandidos Estocásticos, suponemos que los retornos de las acciones están dadas por una función de probabilidad desconocida, cosa que no sabemos si es verdad.
3. En general los algoritmos presentados tratan de elegir qué brazo tiene la mayor esperanza, pero no toman ninguna consideración con respecto a la varianza de cada brazo, es decir no tienen en cuenta el riesgo.
4. No podemos asegurar independencia entre los brazos.
5. No tenemos en cuenta las tasas de corretaje del broker. En un algoritmo que compra y vende mucho, como los presentados, estas tasas podrían comerse todo el beneficio obtenido.
6. Ignoramos los efectos que tendría nuestro algoritmo en el mercado.
7. Suponemos que siempre compramos en el precio de apertura y siempre vendemos en el de cierre, cosa que es imposible de garantizar.

En una primera parte aplicamos los algoritmos de manera directa sobre las acciones. En la segunda parte usamos una segunda acción, además de la elegida por el bandido, con el objetivo de disminuir el riesgo. Y en una tercera parte aplicamos el algoritmo Orthogonal Bandit Portfolio[6].

3.2. Obtención de datos

Todos los experimentos se realizan usando R, los datos se extraen de Yahoo Finance usando la librería de R quantmod. La cual nos permite descargar los datos disponibles de Yahoo Finance a partir del uno de Enero del 2007.

Todos los datos que extraemos, corresponden al periodo desde el 12 de enero de 2007 hasta el 15 de abril del 2016. En esta selección se encuentran algunos eventos de gran importancia como la crisis global del año 2008 y en el caso particular de España, se aprecia los efectos que ocurrieron en el año 2012, cuando la Unión Europea estuvo a punto de rescatar a la economía española, lo que provocó un bajada en los precios de las acciones. El periodo de tiempo elegido para probar, los algoritmos, es semanal. Es decir los retornos se calculan semanalmente.

Cuando seleccionamos las acciones, deseamos que, estas acciones coticen en bolsa en el periodo de tiempo que va desde el 12 de enero de 2007 hasta el 15 de abril del 2016, y eliminamos de nuestro conjunto aquellas acciones que no coticen en bolsa durante ese periodo. Así, por ejemplo Bankia, que empezó a cotizar en bolsa en julio del 2012, no sería un acción válida para nuestras pruebas.

Además en ciertas ocasiones se fusionan o dividen acciones provocando grandes subidas o bajadas del precio de las mismas. Por ejemplo, el 9 de junio de 2014 la compañía Apple realizó un *stock split* sobre sus acciones por el cual una acción de Apple se convirtió en siete acciones de la misma, lo que provocó que su precio se dividiera por siete. El precio de una acción de Apple pasó de valer 645 dólares a 93 dólares. Debido a estos eventos, se generan retornos extraordinariamente grandes o pequeños, que alteran el desarrollo de la prueba. Debido a que el número de acciones que se utilizan en las pruebas es alto, no podemos comprobar en que acciones se producen estos eventos. Por tanto eliminamos de nuestro conjunto aquellas acciones que devuelvan un retorno mayor a un 30 % o un retorno menor a un -30 %.

Usamos un periodo τ de entrenamiento, para que los algoritmos no tengan un inicio en frío. Además usamos una ventana móvil de tamaño también τ para calcular todos los datos estadísticos que necesitemos para la aplicación de nuestros algoritmos. El tamaño de τ que usamos en las pruebas es de 120 semanas. En caso de los retornos semanales esto equivale aproximadamente a dos años y tres meses.

Utilizamos los siguientes grupos de pruebas:

Grupo1. Seleccionamos las 500 acciones del índice *Standard & Poor's 500*, de las cuales sólo nos sirven 225 acciones para nuestras pruebas, según los criterios anteriormente explicados.

Grupo2. Tomamos las 100 acciones del índice *Standard & Poor's 100*, las cuales son las 100 mejores acciones del indice *Standard & Poor's 500*. De este conjunto sólo nos son válidas 59 acciones.

Grupo3. Tomamos 100 acciones aleatorias del índice *Standard & Poor's 500*, de las cuales sólo son válidas 45 acciones.

Grupo4. Tomamos las 35 acciones del índice IBEX 35, de las cuales sólo son válidas 16 acciones.

A la hora, de medir la eficacia de los algoritmos utilizamos las siguientes medidas:

Media muestral de las recompensas obtenidas.

Desviación estándar muestral de las recompensas obtenidas.

Recompensa Acumulada.

Ratio de *Sharpe*, que consiste en la media muestral dividido entre la desviación estándar muestral.

3.3. Aplicación directa de los algoritmos

Los algoritmos que usamos en nuestras pruebas son:

UCB1, el cual ha sido explicado en la Sección 2.7.

El Bayesiano Binomial, en este caso este algoritmo elige el brazo que tenga la mayor probabilidad de dar recompensas positivas. Pero esto puede ser contraproducente, ya que puede existir un brazo que de recompensa positivas a menudo, pero que cuando da recompensas negativas baje mucho.

El Bayesiano Normal, explicado en el Anexo C. Este algoritmo funciona de forma parecida que el bayesiano binomial, sólo que en este caso las recompensas están dadas por funciones de distribución normales.

3.4. Dos acciones

Como se ha indicado anteriormente, uno de los problemas de los algoritmos explicados, es que sólo tienen en cuenta la media a la hora de elegir el brazo. Es por eso, que se hace necesario un mecanismo de disminución del riesgo. Una de las formas más sencillas consiste en la diversificación. Para ello podemos tomar una segunda acción o activo financiero y construimos una cartera conformada por dos activos. Un activo, es el elegido por el algoritmo y el otro, es cualquier otra acción diferente que se encuentre dentro del conjunto de acciones. Una vez que tenemos construido la cartera, usamos el Teorema 6 del Anexo A para disminuir el riesgo.

Por tanto lo que hacemos, es una vez que el algoritmo selecciona un brazo, construimos $n-1$ carteras que tengan la acción elegida y otra acción, los pesos de cada uno de los activos se calculan usando el Teorema 6 del Anexo A, donde n es el número de acciones. Posteriormente calculamos el ratio de *Sharpe* para cada una de las carteras, y elegimos aquella que tenga mayor ratio de *Sharpe*.

Sea n el número de acciones, τ el periodo de entrenamiento, T el periodo de inversión y A uno de los algoritmos vistos anteriormente (UCB1,Bayesiano Binomial o Bayesiano Normal). El algoritmo se puede escribir:

Algorithm 10 Selección Doble

function DOUBLESELECTION(n, τ, T, A)
  for t=1,2,...(T-1) **do**
    El algoritmo A selecciona una determinada acción, I_{t1}
    for i=1,2,...n **do**
      if $i \neq I_{t1}$ **then**
        Calculamos el porfolio del brazo I_{t1} y del brazo i, usando el Teorema 6
        Calculamos el Sharpe Ratio de la cartera, S_i
      end if
    end for
    $I_{t2} = \arg\max_i(S_i)$, por tanto nuestra cartera esta formada por los brazos I_{t1} e I_{t2}
    Todos los brazos devuelven las recompensas, y calculamos el retorno dado por la cartera seleccionada.
  end for
end function

3.5. *Orthogonal Bandit Portfolio*

En esta sección presentamos el algoritmo *Orthogonal Bandit Portfolio*[6] (OBP). En este caso construimos carteras de n activos financieros. En cada tiempo t tenemos un vector de recompensas $R_t = (R_{1,t}, R_{2,t}, ..., R_{n,t})^T$. Para construir una cartera con estos n activos debemos dar pesos a cada uno de estos activos, para ello definimos el vector de pesos $w_t = (w_{1,t}, w_{2,t}, ..., w_{n,t})$. La suma de todos los pesos de la cartera debe ser uno.

$$\sum_{i=1}^{n} w_{i,t} = 1$$

Si $w_{i,t} > 0$ significa que el inversor toma un posición larga en el activo i. Si $w_{i,t} < 0$ entonces el inversor toma un posición corta con respecto al activo i.

Si llamamos Σ_t a la matriz de covarianza dada por los retornos R_t, en el tiempo t. Debido a que la matriz de covarianza es simétrica, entonces es diagonalizable. Además sus autovalores son todos positivos (definida positiva) y sus autovectores son ortogonales entre si. Por tanto la matriz de covarianza se puede escribir como:

$$\Sigma_t = H_t \Lambda_t H_t^T. \tag{3.1}$$

Donde Λ_t es una matriz diagonal que contiene los autovalores de la matriz Σ_t en orden decreciente, $\lambda_{1,t} > \lambda_{2,t} > ... > \lambda_{n,t} > 0$. Y $H_t = (H_{1,t}, H_{2,t}, ..., H_{n,t})$ es una matriz ortonornal cuyas columnas $H_{i,t}$ son los autovectores de la matriz Σ_t. Si multiplicamos $H_t^T R_t$ obtenemos un conjunto de n carteras no correlacionadas, y los autovalores representan la varianza de cada una de estas carteras. Pero el problema es que esas n carteras, sus pesos no suman 1, por tanto debemos normalizar los autovectores de la siguiente forma:

$$\overline{H}_{i,t} = \frac{H_{i,t}}{H_{i,t}^T 1}. \tag{3.2}$$

Donde 1 es el vector unidad y la matriz con los autovectores normalizados quedaría, $\overline{H}_t = (\overline{H}_{1,t}, \overline{H}_{2,t}, ..., \overline{H}_{n,t})$.

Por tanto definimos, un nuevo conjunto de n carteras $\overline{H_t}^T R_t$. La matriz de covarianza de estas n carteras es:

$$\overline{\Sigma}_t = \overline{H}_t \Sigma_t \overline{H}_t^T = \overline{\Lambda}_t. \tag{3.3}$$

Donde $\overline{\Lambda}_t$ es una matriz diagonal cuyo elemento i es: $\overline{\lambda}_{i,t} = \lambda_{i,t}/(H_{i,t}^T\overline{1})^2$, la cual es la varianza de la cartera $\overline{H}_{i,t}^T R_t$. Este nuevo conjunto de carteras las llamamos carteras ortogonales. Lo que hemos hecho hasta aquí es aplicar análisis de los componentes principales, de tal forma que el autovector $H_{1,t}$ indica los movimientos más importantes del mercado, mientras que $H_{n,t}$ indica los movimientos menos representativos del mercado.Por tanto la matriz de covarianza se puede escribir como:

$$\overline{\Sigma}_t = \sum_{i=1}^{l} \overline{\lambda}_{i,t}\overline{H}_{i,t}\overline{H}_{i,t}^T + \sum_{i=l+1}^{n} \overline{\lambda}_{i,t}\overline{H}_{i,t}\overline{H}_{i,t}^T. \tag{3.4}$$

Donde los primeros l factores son movimientos sistemáticos del mercado y los otros $n-l$ factores son movimientos no sistemáticos que los inversores exploran para generar un retorno extra. Dividimos pues, las n carteras no correlacionadas en dos grupos, uno formado por los l movimientos más significativos y otro formado por los $n-l$ movimientos menos significativos. Elegimos l de tal forma que el primer grupo tenga una significancia del 30 % sobre la varianza total. Posteriormente aplicaremos el algoritmo UCB1 a los dos grupos, de tal manera que elegiremos una inversión pasiva de los primeros l *portfolios* y una inversión activa de los $n-l$ *portfolios*.

A la hora de aplicar el algoritmo UCB1, no usamos la media de la cartera si no que calculamos el *Sharpe ratio*, así sólo tenemos en cuenta la media, a la hora de elegir el brazo. El *Sharpe ratio* de cada cartera se calcula como:

$$\overline{r}_{i,t} = \frac{E[\overline{H}_{i,t}R_t]}{\sqrt{\overline{\lambda}_{i,t}}} = \frac{H_{i,t}E[R_t]}{\sqrt{\lambda_{i,t}}}. \tag{3.5}$$

Posteriormente le añadimos el intervalo de confianza del algoritmo UCB1 y elegimos el brazo que tenga la cota superior de confianza mayor. Elegimos dos brazos, uno para cada uno de los dos subconjuntos.

$$i_t^* = \underset{i=1,\dots,l}{\arg\max}\, \overline{r}_{i,t} + \sqrt{\frac{\tau+t}{\tau+n_i}}. \tag{3.6}$$

$$j_t^* = \underset{i=l+1,\dots,n}{\arg\max}\, \overline{r}_{i,t} + \sqrt{\frac{\tau+t}{\tau+n_i}}. \tag{3.7}$$

Donde n_i es el número de veces que ha sido elegido el brazo i y τ es el periodo de entrenamiento. Una vez que tenemos seleccionados los dos brazos, i_t^* y j_t^*, queremos combinar las dos carteras ortogonales seleccionadas en una única cartera, de tal manera que aplicamos el Teorema 6 del anexo C. Si $w_{j,t} = \theta_t$ y $w_{2,t} = 1-\theta_t$, entonces la cartera que contiene estos dos activos financieros tiene la mínima varianza cuando:

$$\theta_t = \frac{\overline{\lambda}_{i_t^+,t}}{\overline{\lambda}_{i_t^+,t} + \overline{\lambda}_{j_t^+,t}}. \tag{3.8}$$

De tal forma que el peso de nuestra cartera en el tiempo t y en la que invertimos es,

$$w_t = (1-\theta_t)\overline{H}_{i_t^*} + \theta_t\overline{H}_{j_t^*}. \tag{3.9}$$

Y la recompensa que obtendríamos en el tiempo t+1 sería, $w_t R_{t+1}$.

Para la estimación de la matriz de covarianza Σ_t, usamos el modelo único de índice explicado en el Anexo B. En cuanto a la estimación de $E[R_t]$, usamos la media muestral de las recompensas. Para el cálculo de estos datos estadísticos usamos una media móvil de tamaño τ. El algoritmo puede escribirse como:

Algorithm 11 Orthogonal Bandit Portfolio

function OTHOGONAL BANDIT PORTFOLIO(n, τ, T)
  for t=1,2,...(T-1) **do**
    Estimamos $E[R_t]$ y Σ_t usando los τ retornos anteriores
    Reslizamos la diagonalización de $\Sigma_t = H_t \Lambda_t H_t^T$
    Normalizamos para obtener los n *potfolios* ortogonales $\overline{\Sigma}_t = \overline{H}_t \Sigma_t \overline{H}_t^T = \overline{\Lambda}_t$
    Calculamos el *Sharpe ratio* de cada cartera ortogonal (3.5)
    Calculamos los intervalos superiores de confianza para cada brazo
    Seleccionar usando el UCB1 dos brazos uno en el grupo de los l movimientos más significativos (3.6) y otro brazo en el grupo de los $n - l$ movimientos menos significativos (3.7).
    Calcular θ_t usando (3.8)
    Calcular los pesos w_t (3.9)
    Las acciones devuelven los retornos R_{t+1}, y calculamos el retorno obtenido por nuestra cartera.
  end for
end function

3.6. Resultados

A continuación mostramos los resultados obtenidos, en los 4 grupos de pruebas, en los que probamos los algoritmos, que se han explicado en esta sección. Además en nuestras pruebas, incluimos también dos carteras de activos consideradas clásicas dentro del mundo financiero, la cartera de mínima varianza (MVP) y la cartera de pesos iguales (EQW).

Cuadro 3.1: Resultados Grupo1, formado por 225 acciones del *Standard & Poor's 500*.

Algoritmo	Media	Des. Estd.	Sharpe Ratio	Recompensa Acumulada
UCB1	$3.29\ 10^{-03}$	$3.74\ 10^{-02}$	$8.78\ 10^{-02}$	1.19
Binomial	$4.25\ 10^{-03}$	$3.23\ 10^{-02}$	$1.31\ 10^{-01}$	1.54
Normal	$8.34\ 10^{-03}$	$5.06\ 10^{-02}$	$1.64\ 10^{-01}$	3.03
DoubleUCB1	$4.81\ 10^{-03}$	$2.86\ 10^{-02}$	$1.67\ 10^{-01}$	1.74
DoubleBinomial	$3.6\ 10^{-03}$	$2.16\ 10^{-02}$	$1.66\ 10^{-01}$	1.31
DoubleNormal	$6.18\ 10^{-03}$	$3.29\ 10^{-02}$	$1.87\ 10^{-01}$	2.24
OBP	$1.17\ 10^{-02}$	$2.17\ 10^{-01}$	$5.41\ 10^{-02}$	4.28
MVP	$2.62\ 10^{-03}$	$1.7\ 10^{-02}$	$1.54\ 10^{-01}$	$9.51\ 10^{-01}$
EWP	$2.93\ 10^{-03}$	$2.07\ 10^{-02}$	$1.41\ 10^{-01}$	1.06

Cuadro 3.2: Resultados Grupo2, formado por 59 acciones que es encuentran entre las 100 mejores del *Standard & Poor's 500*.

Algoritmo	Media	Des. Estd.	Sharpe Ratio	Recompensa Acumulada
UCB1	$2.32\ 10^{-03}$	$3.29\ 10^{-02}$	$7.0\ 10^{-02}$	$8.44\ 10^{-01}$
Binomial	$2.53\ 10^{-03}$	$3\ 10^{-02}$	$8.42\ 10^{-02}$	$9.18\ 10^{-01}$
Normal	$3.36\ 10^{-03}$	$4.1\ 10^{-02}$	$8.19\ 10^{-02}$	1.22
DoubleUCB1	$2.45\ 10^{-03}$	$2.18\ 10^{-02}$	$1.12\ 10^{-01}$	$8.89\ 10^{-01}$
DoubleBinomial	$2.11\ 10^{-03}$	$2.07\ 10^{-02}$	$1.01\ 10^{-01}$	$7.68\ 10^{-01}$
DoubleNormal	$3.25\ 10^{-03}$	$2.55\ 10^{-02}$	$1.27\ 10^{-01}$	1.18
OBP	$1.4\ 10^{-02}$	$2.88\ 10^{-01}$	$4.86\ 10^{-02}$	5.09
MVP	$2.31\ 10^{-03}$	$1.74\ 10^{-02}$	$1.32\ 10^{-01}$	$8.41\ 10^{-01}$
EWP	$2.62\ 10^{-03}$	$2.05\ 10^{-02}$	$1.27\ 10^{-01}$	$9.52\ 10^{-01}$

Cuadro 3.3: Resultados Grupo3 formado por 45 acciones aleatorias del *Standard & Poor's 500*.

Algoritmo	Media	Des. Estd.	Sharpe Ratio	Recompensa Acumulada
UCB1	$-3.11\ 10^{-05}$	$3.55\ 10^{-02}$	$-8.75\ 10^{-04}$	$-1.12\ 10^{-02}$
Binomial	$2.73\ 10^{-03}$	$3.66\ 10^{-02}$	$7.46\ 10^{-02}$	$9.93\ 10^{-01}$
Normal	$2.15\ 10^{-03}$	$4.32\ 10^{-02}$	$4.98\ 10^{-02}$	$7.83\ 10^{-01}$
DoubleUCB1	$2.46\ 10^{-03}$	$2.75\ 10^{-02}$	$8.96\ 10^{-02}$	$8.96\ 10^{-01}$
DoubleBinomial	$2.18\ 10^{-03}$	$2.37\ 10^{-02}$	$9.22\ 10^{-02}$	$7.93\ 10^{-01}$
DoubleNormal	$4.16\ 10^{-03}$	$3.12\ 10^{-02}$	$1.33\ 10^{-01}$	1.51
OBP	$2.06\ 10^{-02}$	$2.18\ 10^{-01}$	$9.42\ 10^{-02}$	7.48
MVP	$2.51\ 10^{-03}$	$1.74\ 10^{-02}$	$1.43\ 10^{-01}$	$9.1\ 10^{-01}$
EWP	$2.87\ 10^{-03}$	$2.12\ 10^{-02}$	$1.35\ 10^{-01}$	1.04

Cuadro 3.4: Resultados Grupo4, formado por 16 acciones del IBEX35

Algoritmo	Media	Des. Estd.	Sharpe Ratio	Recompensa Acumulada
UCB1	$-1.26\ 10^{-03}$	$4.24\ 10^{-02}$	$-2.98\ 10^{-02}$	$-4.6\ 10^{-01}$
Binomial	$1.7\ 10^{-03}$	$3.3\ 10^{-02}$	$5.15\ 10^{-02}$	$6.19\ 10^{-01}$
Normal	$2.03\ 10^{-03}$	$3.52\ 10^{-02}$	$5.79\ 10^{-02}$	$7.4\ 10^{-01}$
DoubleUCB1	$6.78\ 10^{-04}$	$3.21\ 10^{-02}$	$2.11\ 10^{-02}$	$2.46\ 10^{-01}$
DoubleBinomial	$1.65\ 10^{-03}$	$2.72\ 10^{-02}$	$6.087\ 10^{-02}$	$6.01\ 10^{-01}$
DoubleNormal	$1.07\ 10^{-03}$	$3.2\ 10^{-02}$	$3.36\ 10^{-02}$	$3.91\ 10^{-01}$
OBP	$5.56\ 10^{-03}$	$2.67\ 10^{-01}$	$2.08\ 10^{-02}$	2.01
MVP	$2.06\ 10^{-03}$	$3.18\ 10^{-02}$	$6.47\ 10^{-02}$	$7.48\ 10^{-01}$
EWP	$8.17\ 10^{-04}$	$3.3\ 10^{-02}$	$2.47\ 10^{-02}$	$2.96\ 10^{-01}$

En los resultados obtenidos en las pruebas, podemos observar que en general el algoritmo OBP, es el que mejor recompensa acumulada y recompensa media consigue, no obstante tiene

mayor desviación típica si lo comparamos con el resto de los algoritmos. Si bien es cierto que el algoritmo OBP, supera con creces al resto de los algoritmos, esto es así porque los pesos de la cartera suelen incluir números mayores que 1 en aquellas inversiones que tienden a subir y números menores que cero en aquellas inversiones que tienden a bajar. Haciendo que cuando gane, gane mucho, sin embargo cuando se equivoca pierde mucho, lo que genera un desviación estándar grande.

También se puede observar una mejora de la desviación típica de los algoritmos *Double* (DoubleBinomial, DoubleUCB1, DoubleNormal), en relación a la aplicación directa de los algoritmos (UCB1, Normal, Binomial). Se puede observar que se produce una pérdida de recompensa acumulada y recompensa media, sin embargo en el algoritmo UCB1 y en el Grupo 3 (Cuadro 3.3) se produce una mejora de la recompensa acumulada.

Es interesante comprobar que en el Grupo3 (Cuadro 3.3) es donde el algoritmo OBP y los algoritmos *Double* producen los mejores resultados. Esto se podría explicar por la conformación aleatoria del Grupo3. Esta componente aleatoria hace que las acciones tengan una menor correlación que, por ejemplo las acciones del Grupo 2 (Cuadro 3.2).

4

Conclusiones y trabajo futuro

En este trabajo, se ha estudiado el problema del bandido multibrazo, en especial se ha hecho énfasis en los bandidos estocásticos. Son varios los algoritmos estudiados. Posteriormente se han utilizado estos algoritmos en una aplicación práctica como es el diseño de carteras financieras.

Uno de los conceptos más importantes que se han tratado a lo largo de la memoria es el concepto exploración *versus* explotación, se ha visto a lo largo de las múltiples pruebas realizadas que es conveniente explorar al principio de la simulación para ir pasando a la explotación, conforme extraemos información.

Se ha observado que los algoritmos estocásticos eligen los brazos en función de la media muestral de cada uno de ellos, pero no se tiene en consideración la varianza de cada brazo es decir, el riesgo. Se hace necesario por tanto implementar mecanismos con el fin de disminuir el riesgo. Una de las formas más sencillas de disminuir el riesgo es la diversificación. En la sección 3.4 utilizamos una segunda acción para disminuir el riesgo (varianza) de la acción seleccionada por el algoritmo del bandido multibrazo.

Se ha estudiado también el algoritmo OBP el cual crear carteras ortogonales y usa en algoritmo UCB1 para elegir entre estas carteras. A diferencia de los anteriores algoritmos este no elige entre acciones, si no que elige entre diferentes carteras.

Se ha visto que este algoritmo mejora los resultados con respecto a los otros algoritmos presentados en esta memoria, tanto los algoritmos presentados en la Sección 3.3 como los presentados en la Sección 3.4. Si bien es cierto que toma decisiones más arriesgadas a la hora de crear a las carteras, es por ello que tiene un mayor varianza.

Se ha propuesto como método alternativo el bandido bayesiano normal, explicado en el Anexo C. Su funcionamiento es similar al bandido bayesiano explicado en la Sección 2.2 sólo que en este caso las recompensas están dadas por distribuciones normales.

Se ha estudiado además el bandido adversario, se ha incluido en el anexo por tema de limitación de espacio.

Se han implementado todos los algoritmos en lenguaje R y se ha utilizado la biblioteca de R, *quantmod* que permite extraer los datos directamente de *Yahoo Finance*.

En cuanto a trabajos futuros y mejoras, sería interesante el uso de bandidos contextuales para el diseño de carteras. Los bandidos contextuales son aquellos que usan la información del entorno,

no sólo las recompensas devueltas por los brazos, a la hora de elegir un determinado bandido. Además sería también una buena opción usar modelos multifactoriales, para la estimación de la matriz de covarianza, aunque se debería atender a la multicoliniaridad que pudiese existir entre los variables independientes que forman el modelo.

Bibliografía

[1] Andrew G. Barto Richard S. Sutton. *Reinforcement Learning: An Introduction*. A Bradford Book, 1998.

[2] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematics Society*, pages 527–535, 1952.

[3] Sebastien Bubeck and Nicolo Cesa-Bianchi. *Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems*. Now Publishers Inc, 2012.

[4] John Myles White. *Bandit Algorithm for Website Optimization*. O'Reilly, 2012.

[5] Doina Precup Volodymyr Kuleshov. Algorithms for the multi-armed bandit problem. *Journal of Machine Learning Research*, pages 1–48, 2000.

[6] Weiwei Shen Jun Wang Yu-Gang Jiang Hongyuan Zha. Portfolio choices with orthogonal bandit learning. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 974–980, 2015.

[7] John Langford Robert E. Schapire Lihong Li Wei Chu. A contextual-bandit approach to personalized news article recommendation. *World Wide Web Conference*, pages 661–670, 2010.

[8] Cameron Davidson-Pilon. *Bayessian Methods for Hackers*. (Addison-Wesley Data and Analytics, 2015.

[9] Paul Fischer Peter Auer, Nicolo Cesa-Bianchi. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, pages 235–256, 2002.

[10] Peter Auer Nicolo Cesa-Bianchi Yoav Freund Robert E.Saphire. The non-stochastic multi-armed bandit problem. *Annual Symposium on Fundation of Computer Science*, pages 222–231, 2001.

[11] Tomasz Zastawniak Marek Capinski. *Mathematics for Fianace, An Introduction to Fianacial Engineering*. Springer, 2011.

[12] Jianqing Fan Yingying Fan Jinchi Lv. High dimensional covariance matrix estimation using a factor mode. *Journal of Econom trics*, pages 186–197, 2008.

A

Cartera de Dos Activos

A la hora de evaluar un determinado activo financiero, debemos atender a dos factores fundamentales el retorno esperado y el riesgo que comporta dicho activo. Una de las formas de disminuir el riesgo consiste en diversificar el capital entre varios activos financieros. En este anexo analizamos la disminución del riesgo, que obtenemos en una cartera conformada sólo por dos activos, estos métodos son los que se aplican en la Sección 4.

El retorno de una cartera formada por dos activos financieros es:

$$R_p = w_1 R_1 + w_2 R_2$$

Donde R_p, R_1 y R_2 son los retornos de la cartera y de los dos activos financieros respectivamente. Los pesos w_1 y w_2 son la proporción del capital que invertimos en cada uno de los activos y la suma de ambos debe ser uno. Si uno de los pesos es mayor que uno esto implica que el otro peso debe ser negativo(es decir vendemos al descubierto).

La esperanza y la varianza de la cartera serían:

$$E[R_p] = w_1 E[R_1] + w_2 E[R_2]$$

$$Var[R_p] = w_1^2 Var[R_1] + w_2^2 Var[R_2] + 2 w_1 w_2 Cov[R_1, R_2]$$

Nuestro objetivo es minimizar el riesgo, es decir minimizar la varianza de la cartera.

Theorem 4. *Sean $w_2 = s$ y $w_1 = 1 - s$ entonces la cartera con la mínima varianza se obtiene cuando,*

$$s = s_0 = \frac{Var[R_1] - Cov[R1, R2]}{Var[R_1] + Var[R_2] - 2Cov[R1, R2]} \tag{A.1}$$

Este teorema se aplica en la Sección 4 con el fin de disminuir el riesgo. Este teorema lo extraemos del libro de Capinski y Zastawniak[11].

B

Modelo Único de Índice

Uno de los problemas al que nos enfrentamos cuando usamos carteras de un número finito de acciones es la estimación de la matriz de covarianzas. Uno de los métodos que se suelen aplicar, es el uso de modelos multifactoriales[12], en este caso nosotros usamos el modelo único de índice(*Single-Index Model*), que es el modelo más simple que existe, pues utiliza tan sólo una única variable independiente. El modelo puede escribirse:

$$R_{i,t} = \alpha_i + \beta_i R_{M,t} + \epsilon_{i,t}$$

$$\epsilon_{i,t} \sim iidN(0, \sigma^2_{\epsilon,i}), R_{M,t} \sim iidN(\mu_M, \sigma^2_M)$$

$$Cov[R_{M,t}, \epsilon_{i,s}] = 0 \forall t, s$$

Donde $R_{i,t}$ es el retorno de la acción i en el tiempo t y $R_{M,t}$ es el retorno del mercado en el tiempo t. Queremos explicar como se comporta una determinada acción o activo financiero en función del retorno del mercado. El retorno del mercado suele ser el retorno dado por el índice donde se encuentra la acción(SP500, IBEX35, ...), de tal manera que β_i es un indicador del riesgo sistémico de la acción y α_i representa el riesgo no sistémico o diversificable.

Usando el método de los mínimos cuadrados obtenemos que los estimadores de α_i y β_i son:

$$\widehat{\beta}_i = \frac{\widehat{\sigma_{i,M}}}{\widehat{\sigma_M}^2},$$

$$\widehat{\alpha}_i = \widehat{\mu}_i - \widehat{\beta}_i \widehat{\mu}_M.$$

Donde $\widehat{\mu}_i$ y $\widehat{\mu}_M$ es la media muestral de los retornos de las acciones y del mercado respectivamente. Y $\widehat{\sigma}_{i,M}$ es la covarianza muestral de los retornos de la acción y del mercado y $\widehat{\sigma}_M$ es la varianza muestral de los retornos del mercado. El estimador del error es el residuo que es: $\widehat{\epsilon}_{i,t} = R_{i,t} - \widehat{\alpha}_i + \widehat{\beta}_i R_{M,t}$. Y el estimador insesgado de $\sigma^2_{\epsilon,i}$ es:

$$\widehat{\sigma}^2_{\epsilon,i} = \frac{1}{T-2} \sum_{t=1}^{T} \widehat{\epsilon}^2_{i,t}.$$

La varianza de los retornos de las acciones es:

$$Var[R_{i,t}] = \beta_i^2 \sigma^2_M + \sigma^2_{\epsilon,i}.$$

Luego un estimador de la varianza, sustituyendo los valores que hemos obtenido antes sería:

$$\widehat{\sigma}_i = \widehat{\beta}_i^2 \widehat{\sigma}_M^2 + \widehat{\sigma}_{\epsilon,i}^2.$$

La covarianza entre dos retornos de las acciones sería, $Cov[R_i, R_j] = \beta_i \beta_j \sigma_M^2$ luego su estimación quedaría:

$$\widehat{\sigma}_{i,j} = \widehat{\beta}_i \widehat{\beta}_j \widehat{\sigma}_M^2.$$

La estimación de la matriz de covarianza expresada en forma matricial para n acciones quedaría:

$$\widehat{\Sigma} = \widehat{\sigma}_M^2 \widehat{\beta}\widehat{\beta}' + \widehat{D}.$$

Donde:

$$\widehat{\beta} = \begin{pmatrix} \widehat{\beta}_1 \\ \widehat{\beta}_2 \\ \vdots \\ \widehat{\beta}_n \end{pmatrix}, \widehat{D} = \begin{pmatrix} \widehat{\sigma}_{\epsilon,1}^2 & 0 & \cdots & 0 \\ 0 & \widehat{\sigma}_{\epsilon,2}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \widehat{\sigma}_{\epsilon,n}^2 \end{pmatrix}$$

C

Bandido Bayesiano Normal

En este anexo explicamos como obtenemos el bandido bayesiano normal, que utilizamos en la Sección 4. Primero suponemos que los retornos de los brazos están dados por distribuciones normales es decir. $x_i \sim iid\ N(\mu_i, \sigma_i^2)$. Nuestro objetivo es encontrar una distribución a posteriori para μ_i. Para ello suponemos que la distribución a posteriori esta conjugada con a distribución a priori.

Theorem 5. *Si suponemos que la función a priori para una media μ sigue una normal con parámetros μ_0 y σ_0^2 y que conocemos la varianza, σ^2 para la función de verosimilitud, que también sigue una Normal. Entonces la función de distribución a posteriori de la μ sigue también una Normal de parámetros:*

$$E[\mu|x] = (\frac{\mu_0}{\sigma_0^2} + \frac{\sum_{i=1}^{n} x_i}{\sigma^2})/(\frac{1}{\sigma_0} + \frac{n}{\sigma^2}) \tag{C.1}$$

$$Var[\mu|x] = (\frac{1}{\sigma_0} + \frac{n}{\sigma^2})^{-1} \tag{C.2}$$

Primero construimos una distribución a priori para μ_i. Un estimador de μ_i sería la media muestral $\widehat{\mu}_i$, que seguiría distribución:

$$\widehat{\mu}_i \sim N(\mu_i, \sigma_i^2/n)$$

De esta manera construimos la función a priori para la μ_i, sustituimos por la media muestral y la desviación típica muestral en la distribución, de tal manera que la distribución a priori quedaría $N(\widehat{\mu}_i, \widehat{\sigma}_i^2/n)$. Por lo tanto si nos referimos al Teorema 7 $\mu_0 = \widehat{\mu}_i$ y $\sigma_0^2 = \widehat{\sigma}_i^2/n$

Para poder calcular la distribución a posteriori de μ_i tenemos que suponer que conocemos la varianza σ_i^2, para ello estimamos esta varianza usando el modelo único de índice explicado en el Anexo B. Luego en el contexto del Teorema 7, $\sigma^2 = \widehat{\sigma}_M^2 \widehat{\beta}_i^2 + \widehat{\sigma}_{\epsilon,i}^2$.

Al igual que hacíamos en el Binomial, hacemos muestreo de Thompson con la distribución a posteriori a la hora de elegir el brazo.

D

Bandidos Adversarios

D.1. Introducción

En este caso los brazos no dan recompensas según una función de distribución, si no que las recompensas de cada brazo son elegidas por un adversario que elige las recompensas de forma arbitraria o de forma maliciosa, es decir el retorno de las recompensas puede operar de cualquier manera. Podemos suponer en este caso que nos encontramos en un casino amañado. El adversario podría poner todas las recompensas a cero, pero entonces nadie iría al casino. Al igual que en el caso estocástico tenemos K máquinas tragaperras, y en cada tiempo $t \geq 1$ y para cada máquina $i \in 1, ..., K$ el adversario les asigna una recompensa $X_{i,t}$ acotada en $[0, 1]$. Ahora el pronosticador, deberá elegir un brazo $I_t \in 1, ..., K$ y este recibirá una recompensa $X_{I_t,t}$. Para la defición de bandido adversario y de sus distintas nociones de rechazo usamos las definidas por Sebastien Bubeck y Nicolo Cesa-Bianchi[3].

Según como el adversario elija las recompensas cabe distinguir 2 tipos de adversario:

1. Si el mecanismo usado por el adversario, es independiente de las acciones del pronostticador se le denomina adversario incosciente (*oblivious adversary*). Cabe decir que los bandidos estocásticos se pueden incluir dentro de este tipo de bandidos adversarios.

2. Si el mecanismo de selección de recompensas del adversario depende de las acciones del pronosticador, se denomina al adversario como adversario consciente (*non-oblivious adversary*).

Para un número $K > 1$ de maquinas tragaperras y para un horizonte T, podemos escribir el algoritmo como:

Algorithm 12 El Bandido Adversario

```
1: function BandidoAdversario(K, T)
2:     for t=1,2,...T do
3:         El adveversario genera un vector de recompensas X_t = X_{1,t}, ..., X_{K,t}
4:         El algoritmo A elige el brazo I_t ∈ {1...K}
5:         El algoritmo A recibe la recompensa X_{I_t,t}
6:     end for
7: end function
```

Para el algoritmo A definimos la ganancia obtenida del algoritmo A en el horizonte T, es decir las recompensas observadas por el algoritmo A:

$$G_A(T) = \sum_{t=1}^{T} x_{i_t}(t)$$

Donde $i_1, i_2, ..., i_T$ es la secuencia de recompensas dada por el algoritmo A.

Para los bandidos adversarios vamos a estudiar diferentes tipos de rechazos, para una secuencia de recompensas cualesquiera $j_1, j_2, ..., j_T$ definimos la ganancia obtenida por esa secuencia.

$$G_{(j_1,j_2,...,j_T)}(T) = \sum_{t=1}^{T} x_{j_t}(t)$$

Podríamos comparar como de bien se comporta, nuestro algoritmo A con respecto a esa secuencia, definiendo el rechazo con respecto a esa secuencia:

$$G_{(j_1,j_2,...,j_T)}(T) - G_A(T)$$

Debido a que nuestro algoritmo A posiblemente sea aleatorio, sería mejor decir el rechazo esperado con respecto a esa secuencia:

$$G_{(j_1,j_2,...,j_T)}(T) - E[G_A(T)]$$

El peor caso del rechazo esperado del algoritmo A sería:

$$\max_j \{G_j(T) - E[G_A(T)\}$$

Es decir comparamos nuestro algoritmo A con la mejor secuencia posible que se podría obtener para dicho problema. Otra noción más débil del rechazo sería el rechazo débil (*weak regret*).

$$G_{max}(T) - E[G_A(T)] = \max_j \{\sum_{t=1}^{T} x_j(t)\} - E[G_A(T)]$$

Es decir, ahora comparamos el algoritmo A con el mejor brazo, aquel que dentro del horizonte T devuelve la mayor recompensa acumulada.

D.2. EXP3

El algoritmo Exp3[10], es un algoritmo exponencialmente ponderado para exploración-explotación (*Exponential-weight algorithm for Exploration and Exploitation*). Este Algoritmo funciona manteniendo una lista de pesos para cada brazo, los cuales se usan para elegir de forma aleatoria el brazo a pulsar, incrementando los pesos de aquellos brazos que den buenas recompensas. Introducimos además un factor de igualdad $\gamma \in [0,1]$ que determina, que tanto por ciento de la distribución que usamos para elegir los brazos es una distribución uniforme. Si $\gamma = 1$ el algoritmo elegirá los brazos siguiendo una distribución uniforme. Cuanto más grande sea el γ mayor igualdad existe entre los brazos a la hora de ser elegidos. A continuación se describe el algoritmo Exp3:

Algorithm 13 Exp3

function EXP3(K, T, γ)
  Inicialización: $w_i(1) = 1$ for i=1,...,K
  for t=1,2,...T **do**
    $p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K}$ for i=1,...,K
    El algoritmo A elige el brazo I_t con respecto a $p_1(t), p_2(t), ..., p_K(t)$
    El algoritmo A recibe la recompensa $x_{i_t,t} \in [0,1]$
    for j=1,2,...K **do**
      $\widehat{x_j} = \begin{cases} x_j(t)/p_j(t), & \text{if } j = i_t \\ 0, & \text{en otro caso} \end{cases}$
      $w_j(t+1) = w_j(t) exp(\gamma \widehat{x_j}/K)$
    end for
  end for
end function

En cada tiempo t, Exp3 elige una acción con respecto a la distribución $p_1(t), p_2(t), ..., p_K(t)$. Esta distribución es una mezcla entre la distribución exponencial ponderada y la distribución uniforme. Intuitivamente usamos la distribución uniforme para asegurarnos de que el algoritmo Exp3 pruebe todos los brazos.

Cuando el algoritmo Exp3 ha elegido el brazo i_t y ha recibido la recompensa x_{i_t}, calcula la recompensa estimada $\widehat{x_{i_t}} = x_{i_t}/p_{i_t}(t)$. Al dividir la recompensa obtenida por la probabilidad con la que fue obtenida, hacemos que si un brazo tiene una probabilidad muy pequeña de ser elegido, entonces al dividir la recompensa por un número pequeño y menor que 1, aumentamos mucho el tamaño de la recompensas estimada y por tanto hacemos que la probabilidad de volver a elegir este brazo aumente. Además $E[\widehat{x_j}(t)|i_1, i_2, ..., i_{t-1}] = p_j(t)x_j(t)/p_j(t) + (1 - p_j(t))0 = x_j(t)$, donde $i_1, i_2, ..., i_{t-1}$ son la secuencia elegida por el algoritmo Exp3 en los t-1 tiempos anteriores.

El algoritmo Exp3 tiene una cota superior para el rechazo débil.

Theorem 6. *Para cualquier $K > 0$, para cualquier horizonte $T > 0$ y para cualquier $\gamma \in [0,1]$.*

$$G_{max}(T) - E[G_{Exp3}] \leq (e-1)\gamma G_{max} + \frac{Kln(K)}{\gamma} \tag{D.1}$$

proof.

$$W_t = w_1(t) + ... + w_K(t)$$

$$\frac{W_{t+1}}{W_t} = \frac{\sum_{i=1}^{K} w_i(t+1)}{W_t} = \frac{\sum_{i=1}^{K} w_i(t) e^{(\gamma/k)\widehat{x_i(t)}}}{W_t}$$

Como $p_i(t) = (1-\gamma)\frac{w_i(t)}{W_t} + \gamma/K$, entonces:

$$\frac{W_{t+1}}{W_t} = \sum_{i=1}^{K} \frac{p_i(t) - \gamma/K}{1-\gamma} e^{(\gamma/k)\widehat{x_i(t)}}$$

Por Taylor sabemos que $e^x \leq 1 + x + \frac{1}{2}x^2$, luego:

$$\frac{W_{t+1}}{W_t} \leq \sum_{i=1}^{K} \frac{p_i(t) - \gamma/K}{1-\gamma} [1 + (\gamma/K)\widehat{x_i(t)} + (1/2)(\gamma/K)^2 \widehat{x_i(t)}^2]$$

Sabemos que $e = \sum_{k=0}^{\infty} \frac{1}{k!} = 2 + \sum_{k=2}^{\infty} \frac{1}{k!}$, luego $e - 2 = \sum_{k=2}^{\infty} \frac{1}{k!}$, $1/2 = (e-2) - \sum_{k=3}^{\infty} \frac{1}{k!}$, podemos escribir:

$$\frac{W_{t+1}}{W_t} \leq \sum_{i=1}^{K} \frac{p_i(t) - \gamma/K}{1-\gamma} [1 + (\gamma/K)\widehat{x_i(t)} + (e-2)(\gamma/K)^2 \widehat{x_i(t)}^2] \leq$$

$$\leq \sum_{i=1}^{K} \frac{p_i(t) - \gamma/K}{1-\gamma} + \sum_{i=1}^{K} \frac{p_i(t)\widehat{x_i(t)}}{1-\gamma}(\gamma/K) + \frac{(e-2)(\gamma/k)^2}{1-\gamma} \sum_{i=1}^{K} p_i(t)\widehat{x_i(t)}^2$$

Resolvemos los tres sumatorios por separado.

Primer Sumatorio:

$$\sum_{i=1}^{K} \frac{p_i(t) - \gamma/K}{1-\gamma} = \frac{1}{1-\gamma}[\sum_{i=1}^{k}(p_i(t) - \gamma/K)] = \frac{1}{1-\gamma}[\sum_{i=1}^{k}(p_i(t)) - \gamma)] = 1$$

Segundo Sumatorio:

$$\sum_{i=1}^{K} \frac{p_i(t)\widehat{x_i(t)}}{1-\gamma}(\gamma/K) = \frac{\gamma/K}{1-\gamma} \sum_{i=1}^{K} p_i(t)\widehat{x_i(t)} = \frac{\gamma/K}{1-\gamma} x_{i_t}(t)$$

Tercer Sumatorio:

$$\frac{(e-2)(\gamma/k)^2}{1-\gamma} \sum_{i=1}^{K} p_i(t)\widehat{x_i(t)}\widehat{x_i(t)} \leq \frac{(e-2)(\gamma/k)^2}{1-\gamma} \sum_{i=1}^{K} \widehat{x_i(t)}$$

La última desigualdad se consigue porque: $x_{i_t}(t)\widehat{x_{i_t}(t)} \leq \widehat{x_{i_t}(t)} = \sum_{i=1}^{K} \widehat{x_i(t)}$

Luego al final nos queda:

$$\frac{W_{t+1}}{W_t} \leq 1 + \frac{\gamma/K}{1-\gamma} x_{i_t}(t) + \frac{(e-2)(\gamma/k)^2}{1-\gamma} \sum_{i=1}^{K} \widehat{x_i(t)}$$

Tomamos logaritmos en ambos lados de la desigualdad:

$$Log(\frac{W_{t+1}}{W_t}) \leq Log(1 + \frac{\gamma/K}{1-\gamma} x_{i_t}(t) + \frac{(e-2)(\gamma/k)^2}{1-\gamma} \sum_{i=1}^{K} \widehat{x_i(t)})$$

Usando que $1 + a \leq e^a$ obtenemos:

$$Log(W_{t+1}) - Log(W_t) \leq \frac{\gamma/K}{1-\gamma} x_{i_t}(t) + \frac{(e-2)(\gamma/k)^2}{1-\gamma} \sum_{i=1}^{K} \widehat{x_i(t)}$$

Tomamos el sumatorio $\sum_{t=1}^{T}$ en ambos lados y obtenmos que:

$$Log(W_{T+1}) - Log(W_1) \leq \frac{\gamma/K}{1-\gamma} G_{Esp3}(T) + \frac{(e-2)(\gamma/k)^2}{1-\gamma} \sum_{t=1}^{T}\sum_{i=1}^{K} \widehat{x_i(t)}$$

$W_1 = K$ y para un brazo fijado j tenemos que:

$$W_{T+1} \geq w_j(T+1) = w_j(T+1)e^{(\gamma/K)\widehat{x_j(t)}} = \prod_{t=1}^{T} e^{(\gamma/K)\widehat{x_j(t)}} = e^{(\gamma/K)\sum_{j=1}^{T}\widehat{x_j(t)}}$$

Para un brazo fijado j tenemos entonces:

$$(\gamma/K)\sum_{j=1}^{T}\widehat{x_j(t)} - Log(K) \leq \frac{\gamma/K}{1-\gamma} G_{Esp3}(T) + \frac{(e-2)(\gamma/K)^2}{1-\gamma} \sum_{t=1}^{T}\sum_{i=1}^{K} \widehat{x_i(t)}$$

Multiplicamos ambos lados por $(K/\gamma)(1-\gamma)$:

$$(1-\gamma)[\sum_{j=1}^{T}\widehat{x_j(t)} - \frac{KLog(K)}{\gamma}] \leq G_{Esp3}(T) + (e-2)(\gamma/K)\sum_{t=1}^{T}\sum_{i=1}^{K}\widehat{x_i(t)}$$

Luego:

$$G_{Esp3}(T) \geq (1-\gamma)[\sum_{j=1}^{T}\widehat{x_j(t)} - \frac{KLog(K)}{\gamma}] - (e-2)(\gamma/K)\sum_{t=1}^{T}\sum_{i=1}^{K}\widehat{x_i(t)}$$

Tomamos la esperanza en ambos lados.

$$E[G_{Esp3}(T)] \geq (1-\gamma)[\sum_{j=1}^{T}E[\widehat{x_j(t)}] - \frac{KLog(K)}{\gamma}] - (e-2)(\gamma/K)\sum_{t=1}^{T}\sum_{i=1}^{K}E[\widehat{x_i(t)}]$$

Vemos que:$\sum_{t=1}^{T}E[\widehat{x_j(t)}] = \sum_{t=1}^{T}x_j(t) \leq G_{Max}(T)$ y por tanto:

$$E[G_{Esp3}(T)] \geq (1-\gamma)[G_{Max}(T) - \frac{KLog(K)}{\gamma}] - (e-2)(\gamma/K)\sum_{t=1}^{T}\sum_{i=1}^{K}x_i(t)$$

Si aplicamos que: $\sum_{t=1}^{T}\sum_{i=1}^{K}x_i(t) = \sum_{i=1}^{K}\sum_{t=1}^{T}x_i(t) \leq \sum_{i=1}^{K}G_{Max}(T) = KG_{Max}(T)$, obtenemos:

$$E[G_{Esp3}(T)] \geq (1-\gamma)G_{Max}(T) - \frac{KLog(K)}{\gamma} - (e-2)\gamma G_{Max}(T) =$$

$$= G_{Max}(T) - \gamma G_{Max}(T) - \frac{KLog(K)}{\gamma} - (e-2)\gamma G_{Max}(T)$$

$$E[G_{Esp3}(T)] - G_{Max}(T) \geq G_{Max}(T)(-\gamma-(e-2)\gamma) - \frac{KLog(K)}{\gamma} = \gamma G_{Max}(T)(1-e) - \frac{KLog(K)}{\gamma}$$

Multiplicamos por -1.

$$G_{Max}(T) - E[G_{Esp3}(T)] \leq \gamma G_{Max}(T)(e-1) + \frac{KLog(K)}{\gamma}$$

□

Corolarary 7. *Para cualquier $T > 0$, asumiendo que $g > G_{Max}(T)$ y el algoritmo Exp3 se ejecuta con el parametro:*

$$\gamma = \text{mín}\{1, \sqrt{\frac{KLog(K)}{(e-1)g}}\}$$

Entonces:

$$G_{max}(T) - E[G_{Exp3}] \leq 2\sqrt{e-1}\sqrt{gKLog(K)} \leq 2{,}63\sqrt{gKLog(K)} \tag{D.2}$$

Resultados

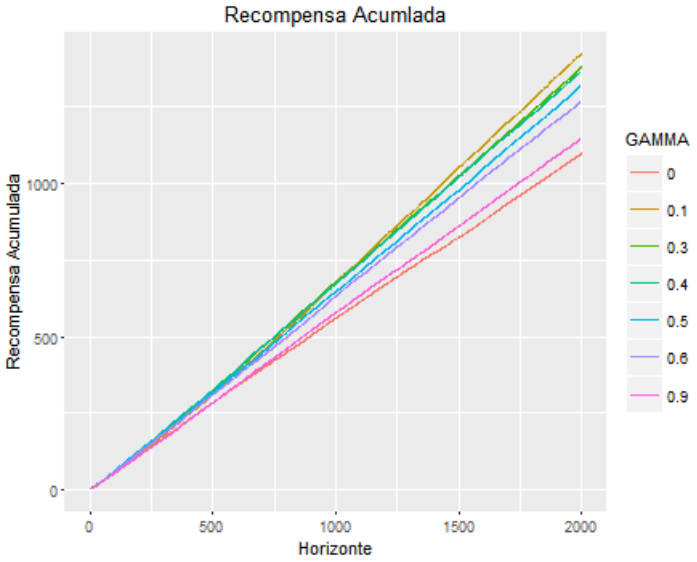

Figura D.1: Recompensa Acumulada del algoritmo Exp3 con seis bandidos Bernuilli con probabilidades 0.3, 0.4, 0.5, 0.6, 0.7 y 0.8. En el eje x se representa el horizonte y en el eje de la y se representa la recompensa acumulada.

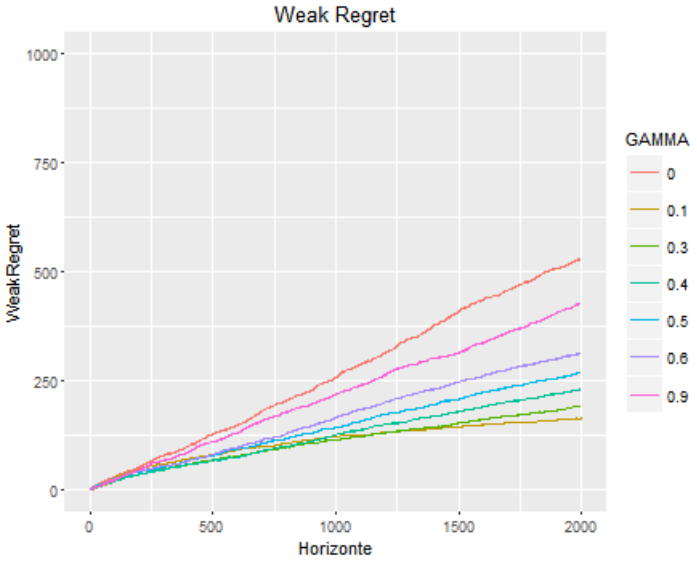

Figura D.2: *Weak Regret* del algoritmo Exp3 con seis bandidos Bernuilli con probabilidades 0.3, 0.4, 0.5, 0.6, 0.7 y 0.8. En el eje x se representa el horizonte y en el eje de la y se representa el rechazo débil.

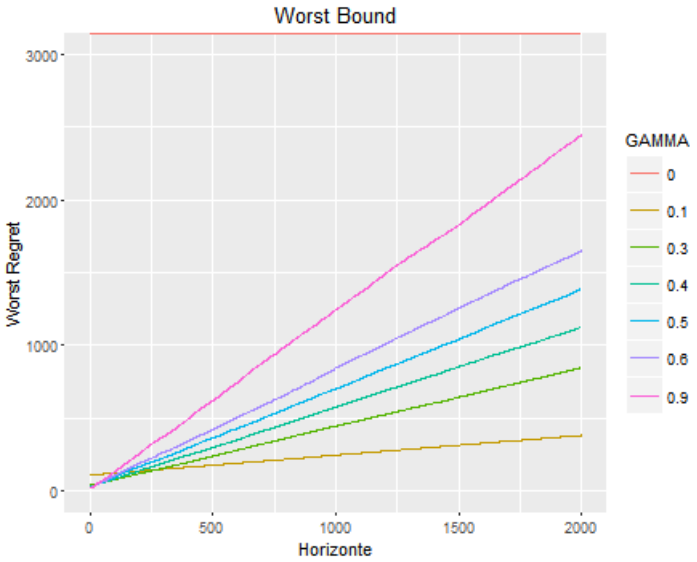

Figura D.3: *Worst Bound* del algoritmo Exp3 con seis bandidos Bernuilli con probabilidades 0.3, 0.4, 0.5, 0.6, 0.7 y 0.8. En el eje x se representa el horizonte y en el eje de la y se representa la cota superior del rechazo débil.

Cuando hablamos de bandidos adversarios conviene usar como medida la recompensa acumulada a la hora de comparar unos algoritmos con otros, aunque en este caso las recompensas vienen dadas por distribuciones Bernuilli, en general las recompensas no tienen porque venir dadas por funciones distribución de probabilidad, por lo que no tiene mucho sentido usar la probabilidad de obtener el mejor brazo, ya que en cada tiempo t el mejor brazo puede ser diferente.

E

Prueba UCB1

Theorem 8. *Para todo $K > 1$, si el algoritmo UCB1 es usado en K máquinas, con distribuciones de recompensas $\nu_1, ..., \nu_K$ acotadas en [0,1] con esperanzas $\mu_1, ..., \mu_k$, entonces el regret para un horizonte T es como mucho:*

$$[8 \sum_{i:\mu_i<\mu^*} log(T)/\Delta_i] + (1 + \Pi^2/3)(\sum_{j=1}^{k} \Delta_j) \tag{E.1}$$

proof. Sea $N_i(T)$ el número de veces que el brazo i ha sido elegido hasta el horizonte T. Vimos que para los bandidos estocásticos el *regret* es:

$$R_T = \sum_{i=i}^{k} \mathbb{E}[N_i(T)]\Delta_i$$

Vamos a acotar $N_i(T)$ sucesivamente para llegar al resultado del teorema.

El bonus $a(j, T) = \sqrt{2log(T)/N_i(T)}$.

Sea I_t es el brazo seleccionado en el tiempo t. Definimos la siguiente función.

$$\chi(A) = \begin{cases} 1, & \text{Si ocurre el evento A} \\ 0, & \text{En caso contrario} \end{cases}$$

Si $\widehat{\mu_i}$ es la media empírica del brazo i. $N_i(T)$ se puede escribir como:

$$N_i(T) = 1 + \sum_{t=K+1}^{T} \chi(I_{t=i})$$

El 1 de esta expresiÃşn proviene de que el algoritmo el UCB1 siempre selecciona 1 vez cada brazo inicialmente. El sumatorio empieza en K+1 porque juega una vez con los K brazos.

$$N_i(T) \leq m + \sum_{t=K+1}^{T} \chi(I_{t=i}, N_i(t-1) \geq m)$$

Suponemos que el brazo ha sido seleccionado al menos m veces, es necesario el menor o igual ya que puede ser que el brazo i haya sido seleccionado menos de m veces.

En general $I_t = i$ cuando $\widehat{\mu}_i + a(N_i(t-1), t-1)$ es el máximo de entre todos los brazos, en este caso nos bastará con que:

$$U_i(t-1) = \widehat{\mu}_i + a(N_i(t-1), t-1) \geq \widehat{\mu^*} + a(N_i^*(t-1), t-1) = U_i^*(t-1)$$

Es decir nos basta con que sea mejor que el bandido óptimo(es decir aquel que tiene la mayor esperanza).Luego podemos escribir que:

$$N_i(T) \leq m + \sum_{t=K+1}^{T} \chi(U_i(t-1) \geq U_i^*(t-1), N_i(t-1) \geq m)$$

$$N_i(T) \leq m + \sum_{t=K+1}^{T} \chi(\max_{m \leq s \leq t} \widehat{\mu_{i,s}} + a(s, t-1) \geq \min_{0<s'<t} \widehat{\mu_{s'}^*} + a(s', t-1))$$

Seguimos agrandando la cota superior.

$$N_i(T) \leq m + \sum_{t=K}^{T} \sum_{s=m}^{t-1} \sum_{s'=1}^{t-1} \chi(\widehat{\mu_{i,s}} + a(s, t-1) \geq \widehat{\mu_{s'}^*} + a(s', t-1))$$

Agrandamos ahora la cota agrandando el primer sumatorio para que vaya desde t=1 a ∞.

$$N_i(T) \leq m + \sum_{t=1}^{\infty} \sum_{s=m}^{t} \sum_{s'=1}^{t-1} \chi(\widehat{\mu_{i,s}} + a(s, t) \geq \widehat{\mu_{s'}^*} + a(s', t))$$

Si $\widehat{\mu_{i,s}} + a(s,t) \geq \widehat{\mu_{s'}^*} + a(s',t)$ entonces al menos uno de estos tres eventos se cumple:

1. $\widehat{\mu_{s'}^*} \leq \mu^* - a(s',t)$
2. $\widehat{\mu_{i,s}} \geq \mu_i + a(s,t)$
3. $\mu^* < \mu_i + 2a(s',t)$

Supongamos que el evento 1. es falso:

$$\widehat{\mu_{i,s}} + a(s,t) \geq \widehat{\mu_{s'}^*} + a(s',t) \leq \mu^* - a(s',t) + a(s',t) = \mu^*$$

Si el evento 2. es flaso:

$$\mu_i + 2a(s,t) > \widehat{\mu_{i,s}} + a(s,t) \geq \widehat{\mu_{s'}^*} + a(s',t)$$

Entonces:

$$\mu_i + 2a(s,t) > \mu^*$$

Luego el evento 3. es verdadero.

El evento 1. ocurre cuando la media empírica del brazo óptimo es menor que su cota de confianza inferior.

El evento 2. ocurre cuando la media empírica del brazo i es mayor que su cota superior.

Por la desigualdad de Chernoff-Hoeffding sabemos que la probabilidades de que ocurran los eventos 1 y 2 respectivamente son:

$$P(\widehat{\mu^*_{s'}} \leq \mu^* - a(s',t)) \leq t^{-4}$$

$$P(\widehat{\mu_{i,s}} \geq \mu_i + a(s,t)) \leq t^{-4}$$

Queremos probar que para un cierto m el evento 3 es imposible que ocurra. Para ello suponemos que todos los eventos son ciertos, por tanto:

$$P(1 \cup 2 \cup 3) \leq 2t^{-4} + P(3)$$

Vamos a ver que si:

$$s \geq m > 8log(T)/\Delta_i$$

Como $s \geq 8log(T)/\Delta_i^2$, entonce se tiene que

$$2a(s,t) \leq \Delta_i^2$$

Si suponemos que 3 es verdadero:

$$\mu^* - \mu_i - 2a(s,t) \geq \mu^* - \mu_i - \Delta_i = 0$$

Por lo que llegamos a contradicción.Por lo que al final obtenemos:

$$E[N_i(t)] \leq m + \sum_{t=1}^{\infty}\sum_{s=m}^{t}\sum_{s'=1}^{t-1} P(\widehat{\mu_{i,s}} + a(s,t) \geq \widehat{\mu^*_{s'}} + a(s',t))$$

$$\leq \lceil 8 * log(T)/\Delta_i^2 \rceil + \sum_{t=1}^{\infty}\sum_{s=1}^{t}\sum_{s'=1}^{t} 2 * t^{-4}$$

$$\leq 8 * log(T)/\Delta_i^2 + 1 + 2\sum_{t=1}^{\infty} t^{-2} = 8 * log(T)/\Delta_i^2 + 1 + \Pi^2/3$$

En la última igualdad aplicamos el problema de Basel que fue resuelto por Euler, $\sum_{n=1}^{\infty} 1/n^2 = \Pi^2/6$.

□